DECISION-THEORETIC ELICITATION OF GENERALIZED ADDITIVE UTILITIES

by

Darius Braziūnas

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Decision-theoretic elicitation

of generalized additive utilities

Darius Braziūnas

Doctor of Philosophy

Graduate Department of Computer Science

University of Toronto

2012

In this thesis, we present a decision-theoretic framework for building decision support systems that incrementally elicit preferences of individual users over multiattribute outcomes and then provide recommendations based on the acquired preference information. By combining decision-theoretically sound modeling with effective *computational* techniques and certain user-centric considerations, we demonstrate the feasibility and potential of *practical* autonomous preference elicitation and recommendation systems.

More concretely, we focus on decision scenarios in which a user can obtain any outcome from a finite set of available outcomes. The outcome is space is *multiattribute*; each outcome can be viewed as an instantiation of a set of attributes with finite domains. The user has preferences over outcomes that can be represented by a utility function. We assume that user preferences are generalized additively independent (GAI), and, therefore, can be represented by a GAI utility function. GAI utilities provide a flexible representation framework for structured preferences over multiattribute outcomes; they are less restrictive and, therefore, more widely applicable than additive utilities. In many decision scenarios with large and complex decision spaces (such as making travel plans or choosing an apartment to rent from thousands of available options), selecting the optimal decision can require a lot of time and effort on the part of the user. Since obtaining the user's complete utility function is generally infeasible, the decision support system has to support recommendation with *partial* preference information.

We provide solutions for effective elicitation of GAI utilities in situations where a probabilistic prior about the user's utility function is available, and in situations where the system's uncertainty about user utilities is represented by maintaining a set of *feasible* user utilities. In the first case, we use Bayesian criteria for decision making and query selection. In the second case, recommendations (and query strategies) are based on the robust minimax regret criterion which recommends the outcome with the smallest maximum regret (with respect to all adversarial instantiations of feasible utility functions).

Our proposed framework is implemented in the UTPREF recommendation system that searches multiattribute product databases using the minimax regret criterion. UTPREF is tested with a study involving 40 users interacting with the system. The study measures the effectiveness of regret-based elicitation, evaluates user comprehension and acceptance of minimax regret, and assesses the relative difficulty of different query types.

Dedication

To my parents, with lots of love and gratitude

Acknowledgements

I would like to thank my advisor Craig Boutilier, my committee members Sam Roweis, Fahiem Bacchus, Sheila McIlraith, Richard Zemel, and the external examiner Yoav Shoham for their time, support and guidance. I am grateful to the faculty and fellow students at the University of Toronto who created a great environment for learning during my graduate studies. Most importantly, it was the love, support and patience of my parents Alma and Vladas, my wife Dilani and my daughter Maya that ensured the completion of this thesis. Thanks!

Contents

List of Tables

List of Figures

Chapter 1

Introduction

One of the main goals of artificial intelligence (AI) research is to devise intelligent systems that help people or companies by recommending or making decisions on their behalf. Application domains for such systems are too many to enumerate, ranging from traditional fields of finance, medicine and logistics to consumer services, such as product recommendation or customization, advertising, and travel planning. The critical requirement for successful implementation of intelligent decision support systems is the knowledge of individual user wants and needs, or *preferences*; once the preferences are known, the system can compute the most preferred course of action based on the available information about the problem domain.

Decision theory provides a formal framework for making optimal decisions by combining probability theory (which describes how actions map to outcomes) with utility theory (which deals with quantitative representation of user preferences). To calculate an optimal decision, both the problem dynamics and user preferences (or *utilities*) over outcomes need to be fully specified. The dynamics of a decision problem are often known and, furthermore, are the same across a variety of users. The opposite is usually the case for user preferences: they vary considerably from user to user, and are rarely fully known. For example, in a travel planning scenario, the distribution over outcomes associated with choosing a specific flight from Toronto to New York (e.g., the odds of a delay greater than one hour, arriving during rush hour, losing

luggage) is the same for any user, but each user's strength of preference for such outcomes can be very different.

While information about the problem dynamics can be inferred from data or specified by domain experts, an intelligent decision support system must be able to extract the preference information from each user individually in order to tailor its actions or recommendations to the specific needs different users. Since obtaining preferences is difficult, this presents a serious problem for the deployment of intelligent systems that make decisions for users with distinct utilities. Effective approaches to *automated preference elicitation* are therefore needed to break the "preference bottleneck", one of the most important problems facing AI.

1.1 Problems and solutions

Successful automated preference elicitation systems must address many difficult challenges. In this section, we identify the key problems, describe previous attempts at solving them, and propose our solutions.

1.1.1 Large outcome spaces

In many realistic domains where the outcome space is large, it is unreasonable to expect a user to provide preference information about every outcome. In multiattribute settings (where outcomes are defined by the possible values of *attributes*) with more than, say, ten attributes, complete preference elicitation becomes virtually impossible, as the number of alternatives grows exponentially in the number of attributes. For example, in a travel planning domain, preferences over the flight selection alone can depend on more than ten attributes, such as trip cost, departure time, return time, airline, number of connections, flight length, baggage weight limit, flight class, the possibility of lost luggage, flight delays, and others. As a different example, which we will also be using throughout this thesis, consider the scenario of searching for a rental accommodation in Toronto. Some of the attributes that strongly influence people's

preferences include the rental price, area, building type, number of bedrooms, proximity to the public transportation, parking availability, and a variety of rental unit features, such as presence or absence of central air-conditioning, balcony, dishwasher, etc. Assessing preferences for all outcomes in either scenario is infeasible due to the size of the outcome space.

User preferences, however, often exhibit internal structure that reflects preferential independencies between domain attributes. *Factored* utility models help overcome the difficulty of preference representation by decomposing user preferences into smaller and more manageable components. The goal of *multiattribute utility theory* (MAUT) (Keeney and Raiffa, 1976) is to investigate numerical representations that reflect structure in user preferences over multiattribute spaces. In MAUT, the outcomes are described by a set of N attributes $X_1, X_2, \ldots, X_N$, each having a finite domain (a set of possible levels, or values). The set of all outcomes (instantiations) $\mathbf{X} = X_1 \times \cdots \times X_N$ is the Cartesian product of attribute levels. The preferences of a user, on whose behalf decisions are made, are captured by a *utility function* $u : \mathbf{X} \mapsto \mathbb{R}$. A quantitative utility function can be viewed as a representation of (qualitative) preferences over *lotteries* (distributions over outcomes), with one lottery preferred to another if and only if its expected utility is greater (von Neumann and Morgenstern, 1947). MAUT describes how, in the presence of certain independence assumptions, user preferences can be represented compactly by an additive, multilinear, or generalized additive utility functions.

Additive independence (Fishburn, 1965; Keeney and Raiffa, 1976) is commonly assumed in practice; the utility u of a multiattribute outcome $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ can then be written as a sum of single-attribute *subutility functions* u_i:

$$u(\mathbf{x}) = \sum_{i=1}^{N} u_i(x_i).$$

In our rental domain, utility of an apartment $\mathbf{y} = (\texttt{downtown}, \texttt{house}, \texttt{2}, \ldots)$ would be decomposed as:

$$u(\mathbf{y}) = u_{area}(\texttt{downtown}) + u_{type}(\texttt{house}) + u_{bedrooms}(\texttt{2}) + \ldots,$$

where u_{area} is the subutility function for the $area$ attribute, and `downtown` denotes its (discrete)

value.

Additive utility models are by far the most widely used due to their simplicity. However, because of the strong preferential independence assumptions, their applicability in many cases might not be justified. An additive decomposition is possible only if user preferences over each attribute are independent of the values of other attributes (a formal definition of additive independence will be provided in the following chapter). In the flight selection scenario, this would imply that the strength of preference over flight length is not influenced by which flight class (business or economy) the user is flying, or the quality of an airline, which is not a realistic approximation. In a rental example, it is quite possible that the preference over the building type does depend on the area; for example, one might prefer an apartment to a house if it is downtown, and a house to an apartment otherwise.

Despite the limitations of additive models, representation and elicitation of more flexible utility models is a relatively unexplored research area.[1] In this thesis, we adopt the *generalized additive independence (GAI)* model (Fishburn, 1967b; Bacchus and Grove, 1995), which decomposes a utility function into subutilities over (overlapping) *sets* of attributes, or *factors*, rather than single attributes.

Let's consider the simplest non-trivial GAI model with three attributes X_1, X_2 and X_3 and two factors $F_1 = \{X_1, X_2\}$, and $F_2 = \{X_2, X_3\}$. In a housing rental scenario, X_1 could be the area, X_2 the building type, and X_3 the number of bedrooms. If, quite realistically, the strength of user preferences over the area (e.g., downtown or not) and number of bedrooms depends on the building type (house or apartment?), then a GAI function with two factors is much more expressive then a simple additive model:

$$u_{GAI}(\mathbf{x}) = u_1(x_1, x_2) + u_2(x_2, x_3).$$

One can intuitively think of GAI factors as a grouping of preferentially dependent attributes

[1] Apart from our work (Braziunas and Boutilier, 2005, 2007), some recent exceptions are (Chajewska, Koller, and Parr, 2000; Boutilier, Bacchus, and Brafman, 2001; Gonzales and Perny, 2004; Boutilier, Patrascu, Poupart, and Schuurmans, 2006; Engel and Wellman, 2007).

(area and building type in one factor, and building type and number of bedrooms in the other).

The GAI representation is completely general—it can represent any utility function—yet in practice is quite intuitive and natural, and typically yields a compact decomposition of preferences. It can model "flat" utility functions with no internal structure as well as additive utilities. Most realistic problems arguably fall somewhere between these two extremes. Due to the generality of the GAI framework, all the results in this thesis are equally applicable to simpler additive functions as well.

1.1.2 GAI utility elicitation

In automated decision support systems, utility models need to facilitate not only representation, but also *elicitation* of user preferences. In addition to ease of representation, simple additive models are attractive because elicitation of additive utility function parameters can be performed by using (almost exclusively) *local* queries. Such queries ask a user for the strength of preference over each attribute in isolation; the values of other attributes are, because of independence, irrelevant. *Global* queries, on the other hand, require consideration of a joint instantiation of *all* attributes, which is cognitively much more difficult for a user if more than a few attributes are involved.

Any additive utility function can be written in a form where an attribute subutility u_i is a product of *local value functions (LVFs)* v_i (normalized to the [0,1] interval) and scaling constants λ_i (all positive, and summing up to 1) (Fishburn, 1965; Keeney and Raiffa, 1976):

$$u(\mathbf{x}) = \sum_{i=1}^{N} u_i(x_i) = \sum_{i=1}^{N} \lambda_i v_i(x_i).$$

This simple factorization separates the representation of preferences into two components: "local" and "global." Since we can define LVFs independently of other attributes, we can also *assess* them independently using local queries involving only the attribute in question. This focus on preferences over individual attributes has tremendous practical significance, because people have difficulty taking into account more than five or six attributes at a time (Green and

Srinivasan, 1978). In our apartment rental example with three attributes, if the user's utility for apartments is given by an additive decomposition, the user only has specify local value functions v_{area}, v_{type}, and $v_{bedrooms}$, and scaling constants λ_{area}, λ_{type} and $\lambda_{bedrooms}$ reflecting the relative utility ranges of each attribute. To assess a single local value function, such as v_{area}, the system need only ask queries involving values of the $area$ attribute, without requiring any consideration of the values of other attributes.

Assessing the scaling constants λ_i cannot be accomplished, of course, without calibration of the LVFs v_i across attributes. This calibration requires the user to compare or evaluate a small number of full outcomes. Fortunately, the number of such outcomes is linear in the number of attributes; the outcomes involved have a very special structure — they each differ from a specific fixed *reference* outcome (typically the worst instantiation of all attributes) by only one feature. The possibility of local assessment makes additive utility the model of choice in most practical applications.

The GAI model is a natural and compact generalization of the much more widely used additive model. While representation of preferences is straightforward, elicitation with GAI models is generally more complicated. Unlike additive models, GAI models require much more care in calibration because of the possible overlap of factors (sharing of attributes). Intuitively, since utility can "flow" from one subutility factor to the next through the shared attributes, the subutility values do not have an independent semantic meaning; unlike in additive models, the values of subutility functions u_i do not directly represent the local preference relation among the attributes in factor i. In addition, there are infinitely many valid decompositions of the same utility function in which the subutility functions vary considerably. Therefore, a *canonical* parameterized representation of the GAI utilities is needed for incremental elicitation of GAI subutilities.

The elicitation issues just described have been largely overlooked in most previous work on GAI utility functions (Boutilier et al., 2001; Boutilier, Patrascu, Poupart, and Schuurmans, 2003b, 2005). One exception is the work by Gonzales and Perny (2004), in which the problem

of local elicitation and global calibration is skirted by using global queries with full (rather than local) outcomes. However, by resorting to full outcome queries, we lose one of the most critical advantages of additive models and fail to exploit the decomposition of utility functions *during the elicitation process*.

We tackle the elicitation issues in GAI models by providing a canonical representation that preserves the local structure of additive models (Braziunas and Boutilier, 2005). A key observation is that by taking into account the *conditioning sets* of attributes that shield the influence of other attribute values on local preferences over factor instantiations (analogously to a Markov blanket in a probabilistic graphical model), we can define semantically sound local value functions for GAI models as well. As in additive models, the LVFs calibrate local preferences relative to the best and worst factor suboutcomes, assuming fixed values of the attributes in the conditioning set. Most importantly, LVFs can be assesed in a local manner by using queries that involve only attributes in single factors and their (usually small) conditioning sets.

Using our representation, GAI models can be elicited by using both local queries about preferences over small subsets of attributes and global queries for calibration across utility factors. We can now exploit the generality, compactness and much wider applicability of GAI models without losing the advantage (offered by more restrictive additive models) of elicitation based on local queries.

1.1.3 Sequential elicitation

Preference information can be obtained in many different ways. "Passive" elicitation methods observe user behaviour (by, for example, examining product buying patterns or search engine query logs) and infer preference models that explain the observed behaviour (Samuelson, 1938, 1948; Louviere, Hensher, and Swait, 2000). "Active" elicitation methods create or refine preference models based on user responses to direct preference queries. In this thesis, we focus on active elicitation. In a sense, we are interested in automating the role of classical decision ana-

lyst whose goal is to assess the user's (or *decision maker's*) utility function through a carefully guided dialogue, and eventually suggest an optimal course of action (Keeney and Raiffa, 1976; French, 1986; French and Insua, 2000).

Active elicitation is by its nature a *sequential process* consisting of a series of questions and answers. Therefore, an automated decision support system must have a policy for query selection (what query to ask next, possibly based on the complete history of previous responses), predefined termination conditions (when to stop the elicitation process), and criteria for making or recommending terminal decisions based on the acquired information. At the core of such an elicitation process is the tradeoff between obtaining more utility information and thus making a better decision, and the cost of further elicitation effort. Elicitation costs can be cognitive (hours of human effort in answering questionnaires), computational (calculating a value of certain alternative might involve solving complicated optimization problems or running simulations), financial (hiring a team of experts to analyze potential business strategies), and others.

A proper accounting for cost vs. decision quality tradeoffs requires an explicit representation of *utility function uncertainty*. Often elicitation costs will outweigh the value of the information obtained by further elicitation, and decisions might have to be made with *partial* utility information. As an example of the tradeoff between decision quality and further elicitation effort, consider the travel planning situation in which, after a number of queries and responses, a decision support system has reduced the set of potentially optimal flights to Flight A, whose utility, after accounting for the ticket price, is between \$800 and \$1000, and Flight B, whose utility is between \$980 and \$1300. Even though it is possible that Flight A is the optimal choice, the maximum possible utility loss, or *regret*, of choosing flight B is only \$20. If the cost of further elicitation is greater than \$20, the system can recommend Flight B without assessing the full utility function.

In this thesis, we consider two ways of representing uncertainty over utility models. The first, Bayesian, approach is to model uncertainty via probability distributions over utility func-

tion parameters; an optimal decision is the one that achieves the highest *expected* utility (where expectation is taken over possible utility parameters). The second approach is to maintain a set of *feasible* utility functions defined by constraints on model parameters. In both cases, we formulate a framework for both making decisions under utility function uncertainty, and elicitation strategies that quickly reduce *relevant* uncertainty.

Bayesian elicitation

In a Bayesian paradigm, utility functions are modeled as random variables drawn from a prior distribution; so, even though the decision support system does not know the user's exact preferences, it has *probabilistic* information regarding her utility function parameters. The value of a decision is computed by taking an expectation over all possible utility functions. The optimal decision is the one with the highest expected utility.

The Bayesian paradigm provides a natural way for determining query strategies as well. As the system obtains more information about user preferences, it incorporates this information into the probabilistic utility model and updates its "beliefs" by applying Bayes's rule. The value of a query can be computed by considering the values of updated belief states (one for each query response), weighted by the probability of corresponding responses. The *expected value of information* (EVOI) is the increase in decision quality that a given query would provide.

The origins of the probabilistic utility uncertainty representation can be traced back to much earlier research in game theory and decision theory: probabilistic modeling of possible payoff functions provides the foundation to the well-established field of Bayesian games (Harsanyi, 1967, 1968); a related concept of *adaptive utility* is discussed in (Cyert and de Groot, 1979; de Groot, 1983); and, Weber (1987) proposes using expectations over utility functions as a possible criterion for decision making with incomplete preference information. Much closer to our approach is the work done in the AI community in the last decade.

In (Chajewska et al., 2000), the authors introduce a Bayesian elicitation framework with *myopic* query strategies. Such a strategy selects a query that greedily maximizes its EVOI

without considering the impact of future queries. Intuitively, at each elicitation step, the query is selected as if it were the last query to be asked before a decision is made. Uncertainty over *unstructured* outcomes is represented by a mixture of Gaussians, which has to be refit after each response because the posterior distribution is not conjugate for the prior for the type of queries employed.

In principle, an optimal elicitation *policy* could be computed *offline* by taking into consideration all possible sequences of questions and answers, providing an optimal tradeoff between query costs (the burden of elicitation) and the potentially better decisions one can make with additional preference information. As shown by Boutilier (2002), finding an optimal sequential elicitation policy amounts to solving a partially observable Markov decision process (POMDP) with a continuous state space. The sequential POMDP model is able to determine the value of a *sequence* of queries, where in isolation, the myopic values of queries in that sequence might be very low. Of course, solving any POMDP is generally computationally difficult, and elicitation POMDP is no exception. The feasibility of this approach has only been demonstrated on very small problems with no multiattribute structure.

Our approach extends the Bayesian paradigm to GAI utility functions over multiattribute outcomes (Braziunas and Boutilier, 2005). Uncertainty is represented by a mixture of uniform distributions over local value function parameters. For computational feasibility, we use a myopic elicitation strategy that employs local bound queries. Because mixtures of uniforms are closed under updates resulting from bound queries, we maintain an exact density over utility parameters throughout the elicitation process. Experimental results show that our approach is fast enough to support interactive real-time preference elicitation on large-size problems (as tested on a 26-attribute domain from (Boutilier et al., 2003b)).

Elicitation under set-based uncertainty

When probabilistic information about user utilities is not available, a decision support system can model the uncertainty by maintaining an explicit representation of the set U of *feasible*

utility functions, viz. those consistent with its knowledge of the user's preferences (e.g., based on responses to previous elicitation queries). We refer to this form of uncertainty as *set-based*, or *strict* (French, 1986), uncertainty. The set U is updated—reduced in size—when new preference information is received during the elicitation process. Unlike the Bayesian case, we cannot say anything about the relative likelihood of the different utility functions in U. If appropriate query types are used, the space U can be conveniently characterized by a collection of *linear* constraints on utility function parameters; if each parameter is bounded, U is simply a *convex polytope* in utility parameter space.

As in the case of Bayesian elicitation, two main issues have to be addressed: how to make decisions given this form of set-based uncertainty, and how to come up with good elicitation strategies. For set-based uncertainty, there is no consensus regarding the best decision criterion. The most common decision rules include maximin return, Laplace's criterion, and minimax regret. The *maximin* criterion (Wald, 1950; Salo and Hämäläinen, 2004) recommends an outcome whose worst-case utility is highest. It is robust, since it provides a minimum utility guarantee, but is often too pessimistic. *Laplace's criterion* simply treats strict uncertainty as uniform uncertainty, and is therefore closer to the Bayesian approach; several non-Bayesian methods (Salo and Hämäläinen, 2001; Iyengar, Lee, and Campbell, 2001; Ghosh and Kalagnanam, 2003; Toubia, Hauser, and Simester, 2004) use this criterion implicitly by recommending a decision that is optimal for the utility polytope *center* (or some approximation thereof).

Under the *minimax regret* decision rule, the system recommends the outcome that minimizes *maximum regret* (or utility loss) with respect to all possible realizations of the user's utility function. First described by Savage (1951) in the context of uncertainty over world states, it has been advocated more recently for robust decision making with incompletely specified utility functions (Boutilier et al., 2001; Salo and Hämäläinen, 2001; Boutilier et al., 2006). Compared to other decision rules, the minimax regret criterion stands out as a reasonable choice because it is intuitive, natural and robust with respect to the worst-case loss under any realiza-

tion of a user's utility function (it provides the tightest bound on such loss).

Minimax regret can be employed to guide the elicitation process as well. With strict uncertainty, many previous approaches (Iyengar et al., 2001; Ghosh and Kalagnanam, 2003; Holloway and White, 2003; Toubia, Simester, Hauser, and Dahan, 2003; Toubia et al., 2004) adopted methods for selecting elicitation queries that could be classified under the general principle of *uncertainty reduction*. The central idea is to choose queries according to criteria based on the size and shape of the feasible utility set. The goal is to reduce the volume of the utility polytope as fast as possible, while also considering the polytope's *shape* (so that uncertainty is "balanced" across multiattribute dimensions). The main limitation of uncertainty reduction methods is the absence of decision quality considerations in elicitation. For instance, one can be quite confident in the quality of a decision despite the fact that considerable uncertainty remains about the user's utility function. The minimax regret criterion does not suffer from this limitation, and effectively drives the elicitation process towards promising regions of the utility space. Each response to a preference query results in a new decision situation inducing a new level of minimax regret (note that regret cannot increase with more preference information). The system can query the user until minimax regret reaches an acceptable level, elicitation costs become too high, or some other termination criterion is met.

Previous work on utility elicitation with the minimax regret criterion has focused on flat utility functions over outcomes with no multiattribute structure (Wang and Boutilier, 2003), additive utilities (Boutilier, Sandholm, and Shields, 2004c), and GAI utilities without semantically sound local queries (Boutilier et al., 2006). In this thesis, we show how to extend minimax regret computation to GAI models with semantically sound local structure, combine several types of both local and global, bound and comparison, queries in elicitation strategies, and achieve fast execution in practice for both constrained configuration problems (where outcomes are defined by hard feasibility constraints over attribute instantiations) and database problems (where outcomes are restricted to those in a finite database set). Our approach is validated by experimental results on synthetic problems (with simulated users) as well as by

an actual user study with 40 participants.

1.1.4 User experience

Interaction with the user is at the core of the automated preference elicitation framework. For many users, queries involving numbers and probabilities are cognitively hard to answer; most users are not experts and therefore require preliminary training. Real case studies often provide evidence of inconsistent responses, errors, and various forms of biases (Simon, 1955; Tversky and Kahneman, 1974; Camerer, Loewenstein, and Rabin, 2003; Pu, Faltings, and Torrens, 2003).

In addition to recommending good options, successful decision support systems must be natural and intuitive, fast (responsive), and efficient (take only as much user time and effort as is necessary). To evaluate different approaches, they have to be implemented in real systems, and tested with real users. Peintner, Viappiani, and Yorke-Smith (2008) provide a recent survey of actual decision support systems. However, few of them have been tested with real users (see (Pu and Chen, 2008) for some exceptions).

To evaluate our GAI-based framework for automated utility elicitation, we implemented the UTPREF recommendation system and tested it in a formal user study with 40 participants. UTPREF is a software tool that explicitly models user preferences with a GAI utility function, incrementally acquires preference information through a sequence of queries and responses, and recommends a minimax regret-optimal option to the user. It employs several types of GAI queries that can be classified as either *comparison* or *bound* queries; queries are further distinguished by the type of outcomes involved (local or global). All query types used have a precise decision-theoretic semantics dictated by the theory of generalized additive independence, and responses to such queries impose linear constraints on the GAI model parameters. An equally important characteristic of these query types is their user-friendliness and usefulness in practical applications.

While very effective in simulated experiments, minimax regret-based elicitation has not

been tested before in realistic domains with real users. We conduct a user study with the UTPREF system with three main objectives: 1) to assess overall user comprehension and acceptance of minimax regret-based elicitation; 2) to measure the costs—in terms of time and perceived difficulty—of different query types; and 3) to evaluate the effectiveness of the GAI utility representation as a model of user preferences, investigate the importance of context in local queries in GAI models, and compare different query strategies. The primary task for the 40 participants was to search for suitable rental apartments, using the UTPREF system, from a database of Toronto apartments. Our results are very encouraging (Braziunas and Boutilier, 2010). We demonstrate that minimax regret is an intuitive, comprehensible decision criterion that can be used to drive effective querying strategies. UTPREF generally produces high-quality recommendations with minimal user preference revelation, and GAI utility models perform better than simple additive models with respect to several recommendation quality measures. Another observation is that simple local queries that omit the local context information perform as well as semantically correct local queries. Finally, the study also allows us to estimate the cognitive costs of different query types, which is very useful in designing better query strategies.

1.2 Contributions

The central task described in this thesis is the development, implementation and evaluation of the comprehensive decision-theoretic framework for automated utility elicitation. Our paradigm casts preference elicitation as an integral part of a sequential process that combines intelligent querying strategies and decision making under utility function uncertainty. Below, we provide our major contributions in a list format:

GAI utility representation

- Identification of problems in earlier methods for GAI utility elicitation

- A new semantically sound representation of GAI utilities that preserves their local structure

- Definition of local value functions as a generalization of LVFs in additive utilities (using the notion of factor conditioning sets)

- An efficient graphical search algorithm for computing GAI structure parameters

- Identification of semantically sound local and global queries for effective elicitation of GAI utility parameters

Bayesian elicitation

- Application of the Bayesian approach to elicitation of local GAI utility parameters

- Design and implementation of a tractable elicitation algorithm using local queries and mixture-of-uniforms priors

- Experimental analysis of the algorithm's performance on large problems using simulated user utilities

Minimax-regret based elicitation

- Tractable formulation for computing minimax regret in GAI models for both configuration and database problems

- MMR-based elicitation of GAI parameters using semantically sound local and global queries

- Experimental validation of the elicitation algorithm

UTPREF recommendation system and user study

- An implementation of the UTPREF recommendation system, the first user-tested system that uses GAI utilities for preference modeling
- The first user study (with 40 participants) to evaluate effectiveness of the minimax regret based approach to utility elicitation, GAI utilities for preference modeling, and costs of different query types.

1.3 Outline

Chapter 2 introduces relevant background material from decision and utility theory (such as decision making under certainty and uncertainty, preference structures over multiattribute outcomes, and factored utility representations), and provides a historical and cross-disciplinary overview of previous approaches to preference elicitation. We concentrate on a few key aspects of the problem: what types of queries can be used to extract user preferences, how to represent utility uncertainty, how to make decisions with partial preference information, and how to devise good elicitation strategies. We survey research fields outside computer science (both historical and current) where preference elicitation plays a central role: imprecisely specified multiattribute utility theory (ISMAUT), its extensions to engineering design and configuration problems, conjoint analysis (in marketing), and analytical hierarchy process (in decision analysis). We also provide an overview of recent attempts to answer the key preference elicitation questions above in the field of AI.

In Chapter 3, we describe the decision-theoretic foundations that support *local* elicitation of GAI utilities and introduce the parameterized representation of GAI utilities that will be used in the remaining chapters. The first part of the chapter is a detailed introduction to GAI utility models, based on the original work by Fishburn (1967b). The second part deals with semantically sound representation of *local* structure in GAI utilities. Local structure facilitates not only representation, but also *elicitation* of utility parameters. In the last part of the chapter,

we present the set of queries for elicitation of GAI utility parameters that we use in our elicitation framework (in both Bayesian and strict uncertainty settings). This chapter is based on the work that first appeared in (Braziunas and Boutilier, 2005).

The foundational material in Chapter 3 applies equally to the following two chapters, which deal with the Bayesian (Chapter 4) and strict (Chapter 5) representation of uncertainty over possible GAI utility functions. In each case, two issues are addressed: 1) how to make good decisions when full utility information is not available; and, 2) what is the best elicitation strategy when further preference information can be obtained. When uncertainty over utilities is quantified probabilistically, Bayesian principles can be applied to both decision making and elicitation of user preferences. In Chapter 4, we propose a mixture-of-uniforms probability model to specify uncertainty over the local parameters of the GAI utility function and use it to perform effective elicitation driven by a myopic EVOI strategy. We conclude the chapter with experimental results on a few large problems with simulated user utilities. Our approach to the Bayesian elicitation of the GAI utilities was first described in (Braziunas and Boutilier, 2005) (and summarized in (Braziunas and Boutilier, 2006)).

Chapter 5 deals with GAI utility elicitation under strict uncertainty. We present tractable algorithms for utility elicitation for configuration (using mixed integer programs) and database (using intelligent search techniques) problems. The minimax regret criterion is used for both decision making and elicitation under utility function uncertainty. We finish the chapter by comparing the performance of various elicitation strategies on several configuration and database problems (with simulated users). The core results in this chapter were published in (Braziunas and Boutilier, 2007).

Chapter 6 concentrates on human-centred issues of utility elicitation. In the first part of the chapter, we present the UTPREF recommendation system that searches multiattribute product databases using the minimax regret criterion. In the second part, we report on a study involving 40 users interacting with the UTPREF system, configured to help users find rental accommodation in Toronto. The study is designed to test the effectiveness of regret-based elic-

itation, evaluate user comprehension and acceptance of minimax regret, and assess the relative difficulty of different query types. The results of the study were published in (Braziunas and Boutilier, 2010).

We conclude in Chapter 7 by providing a summary of the thesis contributions and promising directions for future work.

Chapter 2

Preference elicitation: an overview

Contents

This chapter provides the background material for the thesis, and surveys related work on preference elicitation from a computer scientist's perspective. We consider both historical and current approaches to preference elicitation, concentrating on a few key aspects of the problem. After introducing the relevant concepts and notation of multiattribute utility theory (MAUT), we describe "classical" decision analysis, consider ways to represent uncertainty over utility functions, and summarize various criteria for decision making and query selection with partial utility information. We then survey some research fields where preference elicitation plays a central role. Imprecisely specified multiattribute utility theory (ISMAUT) is one of the earlier attempts to tackle decision making under partial preference information in classical decision analysis. Its extensions to engineering design and configuration problems have been influential in spurring recent interest in preference elicitation among artificial intelligence researchers. Conjoint analysis and analytical hierarchy process (AHP) methods were developed largely in isolation in the fields of marketing research and decision analysis; nonetheless, many issues involving preference elicitation are common. The chapter concludes with a brief overview of recent approaches to preference elicitation in the AI research.

2.1 Decision and utility theory

In this section we provide the background for the decision-theoretic treatment of preferences, including decision making under certainty and uncertainty, preference structures over multiattribute outcomes, and factored utility representations. Decision theory lies at the intersection of many academic disciplines – statistics, economics, psychology, game theory, operations research, and others. Assuming a set of axioms for *rational* behavior, it provides a theory for modeling user preferences and making optimal decisions based on these preferences. The following summary of main concepts is primarily based on the works of von Neumann and Morgenstern (1947); Savage (1954); Fishburn (1970); Keeney and Raiffa (1976); French (1986);

French and Insua (2000).

In the basic formulation of a decision problem, a *decision maker* (DM) has to select a single alternative (or action) a from the set of available alternatives A. An *outcome* (or *consequence*) $x \in X$ of the chosen action a depends on the state of the world $\theta \in \Theta$, also commonly referred to as the *state of nature*. The consequence function $c : A \times \Theta \mapsto X$ maps each action and world state into an outcome. User preferences can be expressed by a *value*, or *utility*, function $v : X \mapsto \mathbb{R}$ that measures desirability of outcomes. The goal is to select an action $a \in A$ that leads to the best outcomes. If the world state θ is known, the set of outcomes is equivalent to the set of alternatives; therefore, in such a case, we will often use these terms interchangeably. If the state θ is unknown (or hidden), a DM has to choose an appropriate *decision criterion* (such a maximin or minimax regret) that determines the procedure for selecting optimal alternatives. When uncertainty over world states is quantified probabilistically, *utility theory* prescribes an action that leads to the highest expected value.

The outcome space itself might be multidimensional. Most interesting problems fall in this category, and we survey some ways of exploiting the structure of utility functions over multidimensional outcomes.

2.1.1 Preferences under certainty

We first consider decisions under certainty. Since the nature state θ is known, each action leads to a certain outcome. Preferences over outcomes completely determine the optimal action: a rational person would choose the action that results in the most preferred outcome.

Let X be a set of outcomes over which a preference relation is defined. The notation $x \succeq y$ means that a person *weakly* prefers outcome x to outcome y; that is, outcome x is deemed to be as good as outcome y. The *weak preference* relation is commonly expected to satisfy the

following two properties for the preferences to be considered *rational*:

$$\begin{aligned} &\text{Comparability:} \quad \forall x, y \in X, \;\; x \succeq y \vee y \succeq x, \\ &\text{Transitivity:} \quad \forall x, y, z \in X, \;\; x \succeq y \wedge y \succeq z \implies x \succeq z. \end{aligned}$$

Weak preference is therefore a *total preorder* (or *weak order*) relation over the set of outcomes X. It is natural to think of weak preference as a combination of strict preference relation $\succ$ and indifference relation $\sim$. The statement $x \succ y$ means that x is strictly preferred to y; $x \sim y$ means that x is exactly as good as y. Formally, for any two elements $x, y \in X$

$$x \sim y \iff x \succeq y \wedge y \succeq x, \tag{2.1}$$

$$x \succ y \iff y \not\succeq x. \tag{2.2}$$

It follows that strict preference is a strict order ($\succ$ is asymmetric and transitive), and indifference is an equivalence relation ($\sim$ is reflexive, symmetric, and transitive).

Weak preferences can be represented compactly by a numerical function. An *ordinal value function* $v : X \mapsto \mathbb{R}$ *represents* or *agrees with* the ordering $\succeq$ when for all $x, y \in X$

$$v(x) \geq v(y) \iff x \succeq y. \tag{2.3}$$

A *representation theorem* gives necessary and sufficient conditions under which some qualitative relation can be represented by a numerical ranking, or *scale*. In case of weak preferences, an agreeing ordinal value function can always be constructed if the outcome set X is finite or countably large. Ordinal value functions are unique up to *strictly increasing transformations*. Such functions are called *ordinal scale* functions. They contain only preference ranking information; if $v(x) \geq v(y)$, we know that x is better (more preferred) than y, but we cannot tell by *how much* it is better. It is meaningless to compare any linear combination of ordinal scale values (such as the average or difference of outcome values).

2.1.2 Preferences under uncertainty

In many settings, the consequences of an action are uncertain. Modern utility theory is based upon the fundamental work of von Neumann and Morgenstern (1947). In this theory, uncertainty is quantified probabilistically, and a rational decision maker is capable of expressing preferences between *lotteries*, or probability distributions over a *finite* set of outcomes.

A *simple lottery* is a probability distribution over outcomes, where outcome x_i is realized with probability p_i. In decision theory literature, it is conventionally denoted as

$$P = \langle p_1, x_1; p_2, x_2; \ldots; p_n, x_n \rangle. \tag{2.4}$$

We will sometimes use $P(x_i) = p_i$ to denote the probability of outcome x_i in the lottery P. It is common to omit outcomes with zero probabilities in the lottery notation.

Let $\mathcal{P}$ be the set of all simple lotteries. If P and Q are two lotteries in $\mathcal{P}$, than any *convex combination* of P and Q is also a simple lottery. That is, for $\alpha \in [0, 1]$, $\alpha P + (1 - \alpha)Q$ is a simple lottery that assigns probability $\alpha P(x_i) + (1 - \alpha)Q(x_i)$ to an outcome x_i. This way of combining simple lotteries can be extended to any finite number lotteries. The result is a *compound lottery* whose outcomes themselves are lotteries. Any compound lottery can be *reduced* to an equivalent simple lottery where the final outcomes are realized with same probabilities. An important assumption about preferences over lotteries is that the decision maker views any compound lottery and its reduction as equivalent; that is, only the ultimate probabilities of outcomes matter. Therefore, it suffices to consider preferences over the set of simple lotteries.

Utility theory axioms

As in the case of certainty, the rational decision maker is assumed to have a complete and transitive preference ranking $\succeq$ over the set of simple lotteries $\mathcal{P}$. The *continuity*, or *Archimedean*,

axiom states that no alternative is infinitely better (or worse) than others:

$$\begin{aligned} \text{Continuity:} \quad & \forall P^1, Q, P^2 \in \mathcal{P}, \\ & P^1 \succ Q \succ P^2 \implies \alpha P^1 + (1-\alpha)P^2 \succ Q \succ \beta P^1 + (1-\beta)P^2, \\ & \text{for some } \alpha, \beta \in (0,1). \end{aligned} \tag{2.5}$$

The continuity axiom is required for existence of a *utility function* $u : \mathcal{P} \mapsto \mathbb{R}$ that represents the preference relation $\succeq$ on simple lotteries. An additional *independence* axiom is necessary to impose a very convenient linear structure on the utility function u:

$$\begin{aligned} \text{Independence:} \quad & \forall P^1, P^2, Q \in \mathcal{P}, \text{ and } \alpha \in (0,1), \\ & P^1 \succ P^2 \implies \alpha P^1 + (1-\alpha)Q \succ \alpha P^2 + (1-\alpha)Q. \end{aligned} \tag{2.6}$$

Independence axiom requires that preferences over P^1 and P^2 carry over to compound lotteries involving some other lottery Q.

The most important result that follows is the *expected utility* representation theorem. It states that if and only if the weak preference relation on simple lotteries is (1) complete, (2) transitive, (3) satisfies the continuity axiom, and (4) satisfies the independence axiom, then there exists an *expected* or *linear* utility function $u : \mathcal{P} \mapsto \mathbb{R}$ which represents $\succeq$. The utility function u has the following properties:

$$\begin{aligned} & (1) \quad u(P) \geq u(Q) \iff P \succeq Q, \\ & (2) \quad u(\alpha P + (1-\alpha)Q) = \alpha u(P) + (1-\alpha)u(Q), \\ & \forall P, Q \in \mathcal{P}, \text{ and } \alpha \in [0,1]. \end{aligned} \tag{2.7}$$

Expected utility theorem

We can identify any outcome $x \in X$ with a degenerate lottery $P^x = \langle 1, x; 0, \ldots \rangle$, where outcome x occurs with certainty. This allows us to extend the preference relation $\succeq$ on simple lotteries to outcomes. The utility of outcome x is then the same as that of the corresponding

degenerate lottery: $u(x) = u(P^x)$. Using induction and linearity of the utility function u, it can be shown that the utility of any simple lottery $P = \langle p_1, x_1; p_2, x_2; \ldots; p_n, x_n \rangle$ is the expected value of its outcomes:

$$u(P) = u(\langle p_1, x_1; p_2, x_2; \ldots; p_n, x_n \rangle) = \sum_{i=1}^{n} p_i u(x_i). \tag{2.8}$$

This key result allows us to represent preferences over an infinite set of simple lotteries by a utility function over a finite set of outcomes.

Utility function uniqueness and standard lotteries

It is a well-known fact that utility functions are *unique up to positive affine transformations* (due to the utility theory axioms and the expected utility theorem). That is, if $u' = au + b$, $a > 0$, then both u' and u represent the same preference relation. Let $x^\top$ be the best outcome, and $x^\perp$ be the worst outcome in X. By setting $a = \frac{1}{u(x^\top) - u(x^\perp)}$, $b = \frac{-u(x^\perp)}{u(x^\top) - u(x^\perp)}$, any utility function u can be transformed to the *normalized* function $u^* = au + b$ whose range is $[0, 1]$ (with $u^*(x^\top) = 1$, $u^*(x^\perp) = 0$).

A *standard lottery* $\langle p, x^\top; (1-p), x^\perp \rangle$ is a simple lottery with positive support on only two special outcomes: the best outcome $x^\top$, and the worst outcome $x^\perp$. Now, assume that a DM is indifferent between some outcome x and the standard lottery $\langle p, x^\top; (1-p), x^\perp \rangle$. Assuming a normalized utility function, we can show that utility of outcome x is p:

$$\begin{aligned} &x \sim \langle p, x^\top; (1-p), x^\perp \rangle \iff \\ &u(x) = u(\langle p, x^\top; (1-p), x^\perp \rangle) = pu(x^\top) + (1-p)u(x^\perp) = p1 + (1-p)0 = p. \end{aligned} \tag{2.9}$$

That is, on the normalized $[0, 1]$ utility scale, $u(x)$ is exactly equal to p, where p is the probability for which x is equally preferred to a lottery in which the best possible outcome $x^\top$ happens with probability p, and the worst outcome $x^\perp$ happens with probability $1 - p$.

A standard lottery is useful in *elicitation* of utilities. In an ideal scenario, since $u(x) = p$, we could assess the utility outcome x by asking the user for the probabability p at which she

would be indifferent between the outcome x for certain, and the standard lottery $\langle p, x^{\top}; (1-p), x^{\perp}\rangle$. In reality, such preference judgments are difficult for humans and subject to noisy responses; thus, in most cases, better ways of obtaining preference information are necessary (see Section 2.2.1.1).

2.1.3 Multiattribute outcomes

In practice, the set of outcomes X is often endowed with multidimensional structure. For example, each alternative in A can be evaluated on several criteria, or *attributes*. Under certainty, action $a \in A$ maps to a point in a *multiattribute* space; under uncertainty, it maps to a distribution over points in that space. The goal of *multiattribute utility theory* (MAUT) is to investigate numerical representations that reflect structure in user preferences over multiattribute spaces.

Assume a set of N attributes $X_1, X_2, \ldots, X_N$. Each attribute has a finite domain (a set of possible levels, or values). For ease of notation, depending on the context, we use X_i to refer both to attributes and their domains. The set of all outcomes (instantiations) $\mathbf{X} = X_1 \times \cdots \times X_N$ is the Cartesian product of attribute levels. Multidimensional outcomes (such as $\mathbf{x} \in \mathbf{X}$) will generally be denoted by lowercase bold symbols.

Given an index set $I \subseteq \{1, \ldots, N\}$, we define $\mathbf{X}_I = \times_{i \in I} X_i$ to be the set of *partial outcomes* (or *suboutcomes*) restricted to attributes indexed by I, and $\mathbf{x}_I$ to be the same restriction applied to some outcome $\mathbf{x}$. $I^C = \{1, \ldots, N\} \setminus I$ is the complement of I.

Example 2.1 Assume a set of three binary attributes X_1, X_2, X_3; each attribute X_i can take one of two values, either x_i^1 or x_i^2. The set of all outcomes $\mathbf{X}$ consists of 8 outcomes: $\mathbf{X} = \{x_1^1 x_2^1 x_3^1,\ x_1^2 x_2^1 x_3^1,\ \ldots, x_1^2 x_2^2 x_3^2\}$. Given an index set $I = \{1, 2\}$, $\mathbf{X}_I$ is a set of partial outcomes, restricted to attributes X_1 and X_2: $\mathbf{X}_I = \{x_1^1 x_2^1, x_1^2 x_2^1, x_1^1 x_2^2, x_1^2 x_2^2\}$.

Preferential independence and the *ceteris paribus* condition

If preferences over multiattribute outcomes exhibit sufficient structure, a preference relation can be modeled more concisely than allowed by the unstructured model discussed in the pre-

vious subsection. In such a case, utility functions can be decomposed into *subutility* functions, defined over subsets of attributes. The simplest *independence condition* is called *preferential independence*. Attributes indexed by I are preferentially independent of the remaining attributes indexed by I^C iff

$$(\mathbf{x}_I, \mathbf{y}) \succeq (\mathbf{x}'_I, \mathbf{y}) \text{ for some } \mathbf{y} \in \mathbf{X}_{I^C} \implies \tag{2.10}$$
$$(\mathbf{x}_I, \mathbf{y}') \succeq (\mathbf{x}'_I, \mathbf{y}') \text{ for all } \mathbf{y}' \in \mathbf{X}_{I^C}.$$

That is, the preferences over $\mathbf{X}_I$, when all other attributes are held fixed, do not depend on the actual values of the remaining attributes. Therefore, a statement $\mathbf{x}_I \succeq \mathbf{x}'_I$, *ceteris paribus* (all else being equal), is a concise way of stating $(\mathbf{x}_I, \mathbf{y}) \succeq (\mathbf{x}'_I, \mathbf{y})$ for all $\mathbf{y} \in \mathbf{X}_{I^C}$.

Ceteris paribus preferential statements provide a natural language for expressing multiattribute preferences. In AI, *ceteris paribus* based assumptions are central to logical models of *qualitative decision theory* (Wellman and Doyle, 1991; Doyle, Shoham, and Wellman, 1991; Doyle and Wellman, 1994; Boutilier, 1994; Tan and Pearl, 1994; Bacchus and Grove, 1996; Doyle and Thomason, 1999).

CP-networks

One issue not addressed by *ceteris paribus* logic theories is that of compact and efficient *representation* of preferential independence statements. *CP-networks*, introduced by Boutilier, Brafman, Hoos, and Poole (1999), provide a popular graphical model that exploits conditional preferential independence among attributes. To create the structure of a CP-net, for each attribute X_i, a user must indicate which other attributes — *parents* of X_i — impact the preferences over values of attribute X_i. Then, for each possible instantiation of the parents of X_i, the user provides a qualitative preference relation over the values of X_i, all else being equal. Given a CP-net, the ceteris paribus semantics induces a *partial* order over full outcomes. Besides providing a compact and natural representation of preferences, a CP-net G can be used to perform preferential comparison between full outcomes ("Does G entail $\mathbf{x} \succ \mathbf{y}$?"), partial

outcome optimization ("What is the best outcome $\mathbf{x}$ given G?"), and outcome ordering ("Is there some ranking in which $\mathbf{x} \succ \mathbf{y}$?").

While partial outcome optimization and outcome ordering are computationally tractable (polynomial in the size of the network), dominance testing is more complicated. In general, answering dominance queries is PSPACE-complete; however, polynomial algorithms exist for tree and polytree structured networks. Another complication is cyclicity: while quite natural in certain settings, cyclical networks are not guaranteed to have a consistent preference ranking. Finally, we should note that introduction of ceteris paribus *indifference* statements can also lead to inconsistent networks.

CP-nets have been extended to deal with hard constraints (Boutilier, Brafman, Domshlak, Hoos, and Poole, 2004a) and quantitative utilities (Boutilier et al., 2001). Another generalization is the TCP-net, which adds conditional importance relations among variables (Brafman and Domshlak, 2002).

Preferences as soft constraints

An alternative, but related, framework for representation and elicitation of multiattribute preferences treats preferences as a "relaxed" form of classical constraints. Solving a *constraint satisfaction problem* (CSP) requires finding an outcome that satisfies all specified "hard" constraints on attribute values (Dechter, 2003). The *soft constraint* framework allows one to assign either discrete or continuous levels of satisfaction to each constraint over a subset of attributes (Prestwich, Rossi, Venable, and Walsh, 2005; Rossi, Venable, and Walsh, 2011). There are many different soft-constraint formalisms (such as weighted constraints, valued constraints, probabilistic constraints, and others), each having different constraint semantics, solution objectives, and solution techniques.

Some recent work within the soft constraint framework considers situations in which some of the soft-constraint values ("preferences") are missing. Gelain, Pini, Rossi, Venable, and Walsh (2010) provide an interactive procedure that interleaves branch-and-bound search with

elicitation steps as it searches for *necessarily* optimal solutions (solutions that are non-dominated under all possible completions of missing values). Rossi and Sperduti (2004) also use an interactive constraint framework, where a user can state constraint preferences as well as preferences over outcomes recommended by the system.

While some problems addressed by the soft-constraint formalisms are closely related to those at the core of automated decision analysis, the underlying semantics of preferences and utilities is very different. In Chapter 5, we will be using some of the CSP techniques to model and solve *constrained optimization* problems.

Pareto-optimal outcomes

Pareto-optimality is an important concept in MAUT under certainty. Consider a set of multi-attribute alternatives A, such that each attribute is preferentially independent of the remaining attributes. Without loss of generality, we can assume that preferences are monotonically increasing with the value of each attribute (for discrete attributes, a notion of Pareto-optimalty requires some natural ordering of attribute values). Then, for $\mathbf{x}, \mathbf{y} \in \mathbf{X}$,

$$\begin{aligned} &\mathbf{x} \textit{ dominates } \mathbf{y} \text{ if} \\ &x_i \succeq y_i \text{ for } i = 1, \ldots, N, \text{ with } x_i \succ y_i \text{ for at least one } i. \end{aligned} \tag{2.11}$$

The *Pareto optimal set* (also known as *efficient set* or *admissible set*) is the set of all nondominated alternatives in A. It is common to focus solely on the Pareto optimal set because a Pareto-dominated alternative cannot be optimal for any utility function or preference relation consistent with local (attribute-specific) preferences.

Utility independence

The idea of preferential independence extends to preferences over lotteries, leading to the notion of *utility independence*. Let $\mathcal{P}$ be the set of all lotteries (probability distributions) on $\mathbf{X}$,

and $\mathcal{P}_I$ be the set of all lotteries on $\mathbf{X}_I$. For $P \in \mathcal{P}$, P_I is the marginal lottery of P over $\mathbf{X}_I$:

$$P_I(\mathbf{x}_I) = \sum_{\mathbf{y} \in \mathbf{X}_{I^C}} P(\mathbf{x}_I\, \mathbf{y}). \tag{2.12}$$

User preferences for $\mathbf{X}_I$ are *utility independent* of $\mathbf{X}_{I^C}$ if preferences for marginal lotteries over $\mathbf{X}_I$ do not depend on the particular assignment to the remaining attributes (those indexed by I^C). Given a marginal lottery P_I, there is a unique lottery $P^{\mathbf{y}}$ such that its marginal distribution over $\mathbf{X}_I$ is P_I, and the probability of outcome $\mathbf{y} \in \mathbf{X}_{I^C}$ is 1:

$$\forall\, \mathbf{x}_I \in \mathbf{X}_I,\, \mathbf{y}' \in \mathbf{X}_{I^C},$$
$$P^{\mathbf{y}}(\mathbf{x}_I\, \mathbf{y}') = \begin{cases} P_I(\mathbf{x}_I), \text{ iff } \mathbf{y}' = \mathbf{y}, \\ 0, \text{ otherwise.} \end{cases}$$

Then, in a way analogous to preferential independence, we can formally define utility independence as follows:

$$P^{\mathbf{y}} \succeq Q^{\mathbf{y}} \text{ for some } \mathbf{y} \in \mathbf{X}_{I^C} \implies \tag{2.13}$$
$$P^{\mathbf{y}'} \succeq Q^{\mathbf{y}'} \text{ for all } \mathbf{y}' \in \mathbf{X}_{I^C}.$$

A graphical *UCP-net* model (Boutilier et al., 2001) that exploits conditional utility independence among attributes is the quantitative analogue of CP-nets (we will discuss some problematic aspects of the UCP-net representation in section 3.2.1.2).

Additive independence

Preferential and utility independence focus on the relationship between two complementary sets of attributes. Additive and generalized additive independence apply to multiple partitions of attributes, either into single attributes or arbitrary sets of attributes. Additive independence relations require much stronger assumptions about the structure of preferences; when such assumptions are justified, additive independence leads to a simple and compact additive utility representation.

Additive independence is a generalization of the utility independence concept. Given a lottery P, let $P_{\{i\}}$ be the marginal lottery on the attribute X_i (as always, we consider N attributes $X_1, \ldots, X_N$).

Definition 2.1 (Additive independence condition) Fishburn (1965)

Attributes $X_1, \ldots, X_N$ are *additively independent* iff

$$(P_{\{1\}}, \ldots, P_{\{N\}}) = (Q_{\{1\}}, \ldots, Q_{\{N\}}) \implies P \sim Q,$$

where $P_{\{i\}} \in \mathcal{P}_{\{i\}}$ denotes the marginal lottery of $P \in \mathcal{P}$ on attribute X_i (for $i = 1, \ldots, N$). That is, the additive independence condition holds if and only if a decision maker is indifferent between two lotteries whenever their marginal distributions on each attribute are the same.

When additive independence holds, preferences can be represented by a utility function u that can be written as a sum of single-attribute *subutility functions* (Fishburn, 1965):

$$u(\mathbf{x}) = \sum_{i=1}^{N} u_i(x_i). \tag{2.14}$$

The additive utility representation also follows if *all* subsets of attributes are mutually preferentially independent of their compliments (Keeney and Raiffa, 1976).

Subutility functions $u_i(x_i)$ are often written as a product of *local value functions* v_i and *scaling constants*, or *weights*, λ_i (often denoted as w_i):

$$u(\mathbf{x}) = \sum_{i=1}^{N} u_i(x_i) = \sum_{i=1}^{N} \lambda_i v_i(x_i), \tag{2.15}$$

where $u_i(x_i) = \lambda_i v_i(x_i)$. The two additive representations — the sum of attribute subutility functions and the sum of weighted local value functions — are equivalent; the weighted representation is almost always used under the assumption that scaling coefficients λ_i form a simplex (i.e., $\sum_i \lambda_i = 1$, $\lambda_i \geq 0$) and local value functions are normalized to be in the range [0,1].

Assume an arbitrary additive function $u(\mathbf{x}) = \sum_i u_i(x_i)$. Let $u^\top = u(\mathbf{x}^\top)$ denote the utility value of the best outcome $\mathbf{x}^\top = (x_1^\top, x_2^\top, \ldots, x_N^\top)$, and $u^\perp = u(\mathbf{x}^\perp)$ to denote the utility

value of the worst outcome $\mathbf{x}^{\perp} = (x_1^{\perp}, x_2^{\perp}, \ldots, x_N^{\perp})$. We use similar notation for subutilities of the best and worst levels of single attributes: $u_i^{\top} = u_i(x_i^{\top})$, and $u_i^{\perp} = u_i(x_i^{\perp})$. As we have shown before, one can always define a strategically-equivalent normalized utility function $u^*(\mathbf{x}) = \frac{u(\mathbf{x})-u^{\perp}}{u^{\top}-u^{\perp}}$. Furthermore, u^* itself is an additive function:

$$u^*(\mathbf{x}) = \frac{u(\mathbf{x}) - u^{\perp}}{u^{\top} - u^{\perp}} = \frac{\sum u_i(x_i) - \sum u_i^{\perp}}{\sum u_i^{\top} - \sum u_i^{\perp}} = \frac{\sum (u_i(x_i) - u_i^{\perp})}{\sum (u_i^{\top} - u_i^{\perp})} = \tag{2.16}$$

$$= \sum \frac{u_i^{\top} - u_i^{\perp}}{\sum (u_i^{\top} - u_i^{\perp})} \frac{u_i(x_i) - u_i^{\perp}}{u_i^{\top} - u_i^{\perp}} \tag{2.17}$$

$$= \sum \lambda_i \, v_i(x_i), \tag{2.18}$$

where $\lambda_i = \frac{u_i^{\top} - u_i^{\perp}}{\sum (u_i^{\top} - u_i^{\perp})}$, and $v_i(x_i) = \frac{u_i(x_i) - u_i^{\perp}}{u_i^{\top} - u_i^{\perp}}$. From the expressions for λ_i and v_i, we can see that $\sum_i \lambda_i = 1$, $0 \leq \lambda_i \leq 1$ and $0 \leq v_i \leq 1$. We should also note that although scaling constants λ_i are often referred to as attribute "weights", they do not reflect attribute importance; rather, they denote relative value ranges of attributes (Fishburn, 1965; Keeney and Raiffa, 1976).

Example 2.2 Table 2.1 shows an example of an additive function, together with its strategically-equivalent normalized representation that uses scaling constants and local value functions.

Linear utility functions

If attributes x_i are numerical and the utility function can be written as

$$u(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i \, x_i, \tag{2.19}$$

then it is *linear*. Such functions are quite commonly assumed in operations research, cost-benefit analysis, and economics. In addition to conditions required for existence of additive value functions, there is an additional property, known as *constant relative tradeoff* between every pair of attributes, that has to be satisfied. A pair of attributes X_i and X_j has a constant relative tradeoff ρ_{ij} if the decision maker is always indifferent between some outcome $\mathbf{x}$ and an outcome obtained by increasing X_i and decreasing X_j in the ratio $\rho_{ij} : 1$. Linear functions are

x_1	$u_1(x_1)$	x_2	$u_2(x_2)$	x_3	$u_3(x_3)$
x_1^1	5	x_2^1	6	x_3^1	0
x_1^2	10	x_2^2	3	x_3^2	7
x_1^3	12	x_2^3	1	x_3^3	2

(a) Additive function u

x_1	$u_1^*(x_1)$	λ_1^*	$v_1^*(x_1)$	x_2	$u_2^*(x_2)$	λ_2^*	$v_2^*(x_2)$	x_3	$u_3^*(x_3)$	λ_3^*	$v_3^*(x_3)$
x_1^1	0		0	x_2^1	0.26		1	x_3^1	0		0
x_1^2	0.26	0.37	0.71	x_2^2	0.11	0.26	0.4	x_3^2	0.37	0.37	1
x_1^3	0.37		1	x_2^3	0		0	x_3^3	0.11		0.29

(b) Normalized additive function u^*

Table 2.1: Table (a) shows an example of an additive utility function $u(\mathbf{x}) = u_1(x_1) + u_2(x_2) + u_3(x_3)$ with three attributes, each having three values. Table (b) shows parameters of a strategically-equivalent normalized utility function $u^*(\mathbf{x}) = u_1^*(x_1) + u_2^*(x_2) + u_3^*(x_3) = \lambda_1^* v_1^*(x_1) + \lambda_2^* v_2^*(x_2) + \lambda_3^* v_3^*(x_3)$.

therefore measured on a more restrictive *ratio* scale: they are unique up to scaling by a positive constant.

Generalized additive independence

While additive models are by far the most commonly used in practice, *generalized additive independence* (GAI) models have recently gained attention because of their additional flexibility (Bacchus and Grove, 1995; Boutilier et al., 2001, 2003b; Gonzales and Perny, 2004; Boutilier et al., 2005; Braziunas and Boutilier, 2005, 2007). The conditions under which a GAI model provides an accurate representation of personal preferences were first defined by Fishburn (1967b, 1970), who introduced the model. GAI is a generalization of the additive model, where independence holds among certain *subsets* of attributes, rather than individual attributes.

Consider a subset of attributes $F_I = \{X_i \mid i \in I\}$ that contains the attributes whose indices

$\mathbf{x}_1$	$u_1(\mathbf{x}_1)$
x^1y^1	100
x^2y^1	70
x^1y^2	90
x^2y^2	50

(a) Factor F_1

$\mathbf{x}_2$	$u_2(\mathbf{x}_2)$
y^1z^1	100
y^2z^1	50
y^1z^2	60
y^2z^2	30

(b) Factor F_2

$\mathbf{x}$	$u_1(\mathbf{x}_1) + u_2(\mathbf{x}_2)$	$u(\mathbf{x})$
$x^1y^1z^1$	100+100	200
$x^2y^1z^1$	70+100	170
$x^1y^2z^1$	90+50	140
$x^2y^2z^1$	50+50	100
$x^1y^1z^2$	100+60	160
$x^2y^1z^2$	70+60	130
$x^1y^2z^2$	90+30	120
$x^2y^2z^2$	50+30	80

(c) Utilities of full outcomes

Table 2.2: An example of a GAI model with three binary attributes X_1, X_2 and X_3 grouped into two GAI factors $F_1 = \{X_1, X_2\}$, and $F_2 = \{X_2, X_3\}$. Therefore, $u(\mathbf{x}) = u_1(\mathbf{x}_1) + u_2(\mathbf{x}_2)$. All instantiations of factor attributes are listed in the first column of Tables 2.2a and 2.2b. The set of all outcomes $\mathbf{X}$ is listed in the first column of Table 2.2c. The second column of Tables 2.2a and 2.2b shows the subutility values of factor outcomes, and the second and third columns of Table 2.2c shows the full utility value of outcomes in $\mathbf{X}$.

are in some index set $I \subseteq \{1, \ldots, N\}$. We will refer to such attribute subsets as *factors*. Assume a collection of M attribute subsets (factors) $\{F_{I_1}, F_{I_2}, \ldots, F_{I_M}\}$ that *cover* the set of all attributes, so that $F_{I_1} \cup F_{I_2} \cdots \cup F_{I_M} = \{X_1, X_2, \ldots, X_N\}$. The factors (and their associated sets of indices) will be commonly enumerated from 1 to M: $F_1, F_2, \ldots, F_M$. In such a case, it will be assumed that $F_j = F_{I_j}$. For a factor F_j, $\mathbf{x}_{I_j}$, or simply $\mathbf{x}_j$, is a particular instantiation of its attributes. Table 2.2 shows an example of GAI model with three attributes and two factors.

Let $\mathcal{P}$ be the set of all lotteries (probability distributions) on $\mathbf{X}$, and $\mathcal{P}_I$ be the set of all lotteries on $\mathbf{X}_I$. As before, for $P \in \mathcal{P}$, P_I is the marginal lottery of P over $\mathbf{X}_I$. By convention, $P_j = P_{I_j}$ will denote the marginal lottery over all the partial outcomes in factor F_j.

Definition 2.2 (Generalized additive independence condition) Fishburn (1967b)
Factors $F_1, \ldots, F_M$ are *(generalized) additively independent* if

$$(P_1, \ldots, P_M) = (Q_1, \ldots, Q_M) \implies P \sim Q,$$

where $P_j, Q_j \in \mathcal{P}_j$ are the j-th marginals of $P, Q \in \mathcal{P}$ (for $j = 1, \ldots, M$). That is, the GAI condition holds if and only if a decision maker is indifferent between two lotteries whenever their marginal distributions on $\mathbf{X}_{I_1}, \ldots, \mathbf{X}_{I_M}$ are the same.

When generalized additive independence holds, the utility of a multiattribute outcome can be written as a sum of subutilities involving GAI subsets of attributes:

$$u(\mathbf{x}) = \sum_{i=1}^{M} u_i(\mathbf{x}_{I_i}). \tag{2.20}$$

Example 2.3 Let's consider the simplest non-trivial GAI model with three attributes X_1, X_2 and X_3 grouped into two GAI factors indexed by $I_1 = \{1, 2\}$, and $I_2 = \{2, 3\}$ (similar to the example in Table 2.2). In a travel planning scenario described in the introduction, X_1 could be the airline, X_2 the flight class, and X_3 the flight length. If, quite realistically, the strength of user preferences over flight class (e.g., economy or business) and airline depends on the flight length, then the simple additive user utility function

$$u_A(\mathbf{x}) = u_1(x_1) + u_2(x_2) + u_3(x_3)$$

cannot adequately represent her true preferences. In contrast, a GAI function with two factors can express such conditional preferences:

$$u_{GAI}(\mathbf{x}) = u_1(x_1, x_2) + u_2(x_2, x_3).$$

GAI models will be the central subject of this thesis.

Other factored utility representations

Several other quantitative representations of preferences over multiattribute outcomes have been proposed in the decision analysis and AI literature. In addition to additive utility functions, Keeney and Raiffa (1976) discuss conditions under which user preferences can be expressed by *multilinear* or *multiplicative* utility functions. In AI, Bacchus and Grove (1995) explore the graphical utility representations of conditional additive independence (including GAI models), Boutilier et al. (2001) propose UCP networks that combine GAI models with CP-net semantics, and Engel and Wellman (2008a) explore a weaker notion of *conditional utility independence* to define CUI networks. La Mura and Shoham (1999) use a different notion of utility independence to introduce *expected utility networks* as undirected graphs that contain both utility and probability dependence information.

2.2 Main aspects of preference elicitation

The increased interest in automated decision support tools in recent years has brought the problem of *automated preference elicitation* to the forefront of research in decision analysis (Dyer, 1972; White, Sage, and Dozono, 1984; Salo and Hämäläinen, 2001) and AI (Chajewska, Getoor, Norman, and Shahar, 1998; Chajewska et al., 2000; Boutilier, 2002). The goal of automated preference elicitation is to devise algorithmic techniques that will guide a user through an appropriate sequence of queries or interactions and determine enough preference information to make a good or optimal decision.

In this section, we concentrate on a few key aspects of the preference elicitation problem. The previous section dealt with various complete representations of preference information. Here, we are interested in the actual process of acquiring such information as well as making decisions with partially elicited utility functions. Therefore, we address issues of how to represent uncertainty over possible utility functions, how to make decisions without full knowledge of user preferences, and how to intelligently guide the elicitation process by taking into account the cost of interaction and potential improvement of decision quality.

2.2.1 "Classical" preference elicitation

Preference (or utility) elicitation is a process of assessing preferences or, more specifically, utility functions. Utility elicitation literature is as old as utility theory itself; first attempts to describe procedures for evaluating utility functions date back to the 1950s. In the "classical" setting, a *decision analyst's* task is to help elicit a *decision maker's* (DM) preferences. Once those preferences are extracted, the decision analyst calculates an optimal course of action according to the utility theory, and *recommends* it to the decision maker (Keeney and Raiffa, 1976; Howard and Matheson, 1984; French, 1986).

There are many techniques for evaluating utility functions, and the whole process of elicitation is "as much of an art as it is a science" (Keeney and Raiffa, 1976). A classical approach, involving an interaction between the decision analyst and the decision maker, usually consists of five steps (Keeney and Raiffa, 1976; Farquhar, 1984). During the *preparation for assessment*, the DM is acquainted with the decision problem, possible outcomes or attributes, and various aspects of the elicitation procedure. The next stage is *identification of relevant qualitative characteristics* of DM's preferences. This could include determining the properties of the utility function (such as continuity or monotonicity in case of numerical attributes), best and worst outcomes or attribute levels, and independence relations among attributes for structured outcome spaces. The central stage is *specification of quantitative restrictions* and *selection of a utility function*. Here, the decision analyst asks various queries, some of which are described

below, in an attempt to model DM's preferences by a *completely* specified utility function. Most of the approaches described in this survey depart from the classical form of elicitation because of the complexity of this task. The last step usually involves *checks for consistency* and *sensitivity analysis*. When inconsistencies are detected, the DM is asked to revise her preferences. The goal of sensitivity analysis is to check the sensitivity of the output (which, in most cases, is the decision recommended by the decision analyst) to the inputs — the utility model and DM's responses.

In the user study described in Chapter 6, elicitation sessions follow roughly the same basic pattern described here.

2.2.1.1 Query types

The nature of queries is an integral part of the preference elicitation problem. Some queries are easy to answer, but do not provide much information; and, vice versa, informative queries are often costly. Another tradeoff to consider is the complexity of selecting a good query versus its potential usefulness. Such aspects of preference elicitation depend on *query types*.

Here, we survey some queries that are commonly used in decision analysis and describe their main characteristics. "Global" queries are applicable to situations where either the set of outcomes does not have any structure, or, in case of multiattribute problems, that structure is ignored and only full, or *global*, outcomes are considered. In most multiattribute problems, people can meaningfully compare outcomes with no more than five or six attributes (Green and Srinivasan, 1978). Therefore, most of the global queries have "local" counterparts that apply to a subset of attributes. We assume that preferences over the set of outcomes X can be expressed by a utility function u, and consider queries that help assess this function.

Order comparison *Order queries* are very simple queries that ask the user to compare a pair of alternatives x and y; the user might prefer x to y, y to x, or be indifferent between the two. Such queries are very common in practice and usually require little cognitive effort from the

user. They are central in ISMAUT and conjoint analysis (Green and Rao, 1971; White, Dozono, and Scherer, 1983; White et al., 1984). Unfortunately, often they are not very informative.

More complicated comparison queries ask the user to pick the most preferred alternative from the set of k alternatives. This rather easy task actually provides $k-1$ pairwise preference comparisons (the selected alternative is preferred to all remaining choices), and is widely used in choice-based conjoint analysis (Louviere et al., 2000; Toubia et al., 2004). At the most extreme, a *total ranking* query expects the user to rank all specified alternatives; answering such a query would provide preference information relating every pair of alternatives.

Most other utility elicitation queries involve simple lotteries, or *gambles*, with only two outcomes. Let $\langle x, p; y\rangle$ be a lottery where x occurs with probability p and y occurs with probability $1-p$. We consider a general query expression $\langle x, p; y\rangle \gtrless z$, where everything except one item is specified, and the user is asked to provide the value of the item that would make the expression true. In the expression, x, y, z are outcomes in X, p is a probability, and $\gtrless$ is either $\succeq$ or $\prec$.[1] The following terminology and classification follows (Farquhar, 1984), who describes queries for all possible combinations of known and unknown quantities in the query expression, as well as more general queries involving two gambles.

Probability equivalence Probability equivalence queries elicit an indifference probability p for which $\langle x, p; y\rangle \sim z$. In a standard gamble case, when $x = x^{\top}$ and $y = x^{\perp}$, the query simply asks to specify the utility of z, and is therefore sometimes called a *direct utility query*. While such queries have been used in research papers (Keeney and Raiffa, 1976; Gonzales and Perny, 2004), it is unlikely that users can provide exact utility values for outcomes in real-world situations. One possible generalization is to ask for *bounds* on the utility value; these bounds can then be narrowed by asking binary comparison queries described below (Boutilier et al., 2003b).

[1] Or, more generally,one of the three preference relations $\succ$, $\sim$ and $\prec$.

Preference comparison In a preference comparison between a gamble $\langle x, p; y\rangle$ and a sure outcome z, a user is asked to specify a relation ($\succeq$ or $\prec$) that holds between the two. When the gamble is a standard gamble, $u(\langle x^{\top}, p; x^{\perp}\rangle) = p$, and the query becomes equivalent to "Is $u(z) \geq p$?" with possible $\{yes, no\}$ responses. Such query is called a *standard gamble comparison query*, or *bound query*, since after the response, p becomes either the upper or lower bound on $u(z)$.

Standard gamble comparison queries are common in classical decision analysis literature (Keeney and Raiffa, 1976). More recently, such queries have been used by Chajewska and Koller (2000); Boutilier (2002); Wang and Boutilier (2003); Boutilier et al. (2003b, 2005); Braziunas and Boutilier (2005, 2007, 2010), and others.

Implicit queries Until now, we assumed that a query is an explicit question posed by the decision support system, and a response is a user's reaction to the query. However, queries and responses can be much more general. The system could pose "implicit" queries by changing the user environment (such as options available on the web page), and observing the user's behavior (links followed, time spent on the page, etc.). Or, the user can be asked to view a fragment of some action policy, and asked to critique the actions. A related theoretical framework is *inverse reinforcement learning* (Ng and Russell, 2000; Chajewska, Koller, and Ormoneit, 2001). The goal is to recover the reward function (preferences) of an agent by observing execution of an optimal policy.

The concept of *revealed preference* in economics (Mas-Colell, Whinston, and Green, 1995) is also related to the topic of implicit queries. Here, the emphasis is on descriptive, rather than prescriptive, aspects of human decision making. Observable choices that people make faced with an economic decision provide the primary basis for modeling their behavior. A preference relation does not exist *a priori*, but could be *derived* (or revealed) given observed choices that follow certain axioms of rationality.

2.2.1.2 Multiattribute elicitation

All the techniques described above could be also employed to simplify utility elicitation of structured outcomes. To illustrate the main concepts, we consider the case of eliciting an additive utility function

$$u(\mathbf{x}) = \sum_{i=1}^{n} u_i(x_i) = \sum_{i=1}^{n} \lambda_i v_i(x_i), \tag{2.21}$$

where $u_i(x_i)$ are subutility functions that can be written as a product of *local value functions* v_i and *scaling factors*, or *weights*, λ_i.

The assumed utility independence among attributes allows elicitation to proceed *locally*: specifically, each v_i can be elicited independently of other attribute values using any of the techniques described above. Since attributes are preferentially independent, each attribute's best and worst levels (we shall call them *anchor levels*) can be determined separately. Let $x_i^\top$ and $x_i^\bot$ denote the best and worst levels, respectively, of attribute i. Local value functions v_i can be determined by locally measuring values of attribute levels with respect to the two anchor levels. For example, one might use local standard gamble queries to assess $v_i(x_i)$ by asking the user for the probability p for which x_i and the *local* standard lottery $\langle p, x_i^\top;\ 1-p, x_i^\bot\rangle$ would be equally preferred, *ceteris paribus*.

What remains is to bring all the local value scales to the common global utility scale. Essentially, we need to find the true utility of all "anchor" outcomes $x_i^\top$ and $x_i^\bot$ relative to some reference outcome $\mathbf{x}^0$ (it is customary to choose the worst outcome as default outcome, and set its utility to 0). Then, eliciting $u(x_i^\top, \mathbf{x}^0_{\{i\}^C}) = u_i(x_i^\top)$ and $u(x_i^\bot, \mathbf{x}^0_{\{i\}^C}) = u_i(x_i^\bot)$ for all attributes would ensure consistent scaling of subutility functions. Because additive utility functions are unique up to positive affine transformations, it is usually assumed that both the global utility functions and local value functions are scaled to lie between 0 and 1; the scaling constants (weights) are also normalized such that $\lambda_i \geq 0$ and $\sum \lambda_i = 1$. In a normalized additive utility function, scaling factors λ_i, which reflect attribute contributions to the overall utility function, are simply equal to $u_i(x_i^\top)$.

The scaling factors can be determined by asking utility queries involving full outcome lotteries. The simplest way to elicit λ_i is to find the utility of $(x_i^\top, \mathbf{x}^\perp_{\{i\}^C})$ for all i:

$$\lambda_i = u(x_i^\top, \mathbf{x}^\perp_{\{i\}^C}). \tag{2.22}$$

In this case, relevant utilities could be elicited by using *global* standard gamble queries. For example, we could ask for the probability p for which the user is indifferent between the outcome $(x_i^\top, \mathbf{x}^\perp_{\{i\}^C})$ and the global standard gamble $\langle p, \mathbf{x}^\top; 1-p, \mathbf{x}^\perp \rangle$. Then, $\lambda_i = u(x_i^\top, \mathbf{x}^\perp_{\{i\}^C}) = p$. Of course, more realistically, instead of asking for the probability p directly, we could use a sequence of binary standard gamble bound queries (or some other less direct queries) to assess the value of p.

More general multiattribute utility functions, such as multiplicative or GAI, can also be elicited using ideas of local value function elicitation and global scaling (see Chapter 3 and previous work by Fishburn (1967a); Keeney and Raiffa (1976); Fishburn (1977)).

2.2.1.3 Problems with the classical paradigm

Complete preference information is rarely attainable in practice. In many realistic domains where the outcome space is large, it is unreasonable to expect a user to provide preference information about every outcome. In multiattribute settings with more than, say, ten attributes, complete preference elicitation becomes difficult, as the number of alternatives is exponential in the number of attributes.

Elicitation of quantitative utilities brings additional difficulties. Queries involving numbers and probabilities are cognitively hard to answer; most users are not experts and therefore require preliminary training. Real case studies often provide evidence of inconsistent responses, errors, and various forms of biases (Simon, 1955; Tversky and Kahneman, 1974; Camerer et al., 2003; Pu et al., 2003). Eliciting preferences might be costly, too; costs can be cognitive (hours of human effort in answering questionnaires), computational (calculating a value of certain alternative might involve solving complicated optimization problems or running simulations), financial (hiring a team of experts to analyze potential business strategies), and others.

Furthermore, from the AI perspective, preference elicitation presents a "bottleneck" for designing automated decision aids ranging from critical financial, medical, and logistics domains to low-stakes processes, such as product recommendation or automated software configuration. For making optimal decisions, we need to know both the decision dynamics and outcome utilities. In many situations, the dynamics is known (representation, elicitation and learning of complex probability models is a well-researched area of AI). However, user preferences are often unknown, and, furthermore, they vary considerably from user to user (while the system dynamics is often fixed for all users).

When costs of elicitation are taken in to account, it becomes clear that decisions might have to be made with partial preference information, if elicitation costs start to exceed the value of potential improvement of decisions. Viewing utility elicitation as an integral part of the decision process is a promising paradigm for tackling the preference elicitation problem.

2.2.2 Decisions with partial preference information

If the utility function is not fully known and further elicitation not possible, what criteria should be used for making good decisions with available information? It turns out that criteria proposed for dealing with state uncertainty in classical decision theory (such as maximum expected utility, minimax regret, maximin) can be applied to situations where utility functions themselves are uncertain. The analogy extends to both common representations of utility function uncertainty: Bayesian, where we can keep track of the probability distribution over possible utility functions, and strict uncertainty, defined by the set of *feasible* utility functions.

2.2.2.1 Strict uncertainty

Under strict uncertainty, knowledge about a user's utility function is characterized by the feasible utility set U. This set is updated (reduced) when relevant preference information is received during an elicitation process. The following is a non-exhaustive list of decision criteria that could be used for making decisions with partial utility information under strict uncertainty.

The set of outcomes is X, and the goal is to choose the best alternative x^* when the set of feasible utility functions is U.

Maximin return Without distributional information about the set of possible utility functions U, it might seem reasonable to select an outcome whose worst-case return is highest:

$$x^* = \arg\max_{x \in X} \min_{u \in U} u(x). \tag{2.23}$$

Maximin decision is sometimes called *robust* because it provides an *ex post* security guarantee. Maximin was proposed by Wald (1950), and mentioned by Salo and Hämäläinen (2004) for the case of uncertain utilities.

Hurwicz's optimism-pessimism index Maximin return is a pessimistic criterion, because the decision maker prepares for the worst realization of the utility function. *Maximax return* criterion is the optimistic counterpart to maximin. Supposing that maximin and maximax are too extreme, Hurwicz proposed to use a weighted combination of the minimum and maximum possible values (French, 1986). For the case of strict uncertainty over utility functions, this criterion would choose

$$x^* = \arg\max_{x \in X} \left[\alpha \min_{u \in U} u(x) + (1 - \alpha) \max_{u \in U} u(x) \right], \tag{2.24}$$

where α is the *optimism-pessimism index* of the decision maker. Hurwicz's optimism-pessimism criterion generalizes minimax ($\alpha = 1$) and maximax ($\alpha = 0$), as well as the *central values* criterion favored by Salo and Hämäläinen (2001). The central values rule prescribes an outcome whose average of its minimum and maximum possible values is highest. This is equivalent to setting the optimism-pessimism index α to 0.5.

Minimax regret Minimax regret criterion was first described by Savage (1951) in the context of uncertainty over world states, and advocated by Boutilier et al. (2001) and Salo and Hämäläinen (2001) for robust decision making with uncertain utility functions. The main idea

is to compare decisions for *each* state of uncertainty. The (pairwise) *regret* of choosing outcome x instead of y is $R(x, y; U) = \max_{u \in U} u(y) - u(x)$. The *maximum regret* of choosing x is $MR(x; U) = \max_{u \in U} \max_{y \in X} [u(y) - u(x)]$. The minimax regret optimal decision minimizes the worst-case loss with respect to possible realizations of the utility function:

$$x^* = \operatorname*{argmin}_{x \in X} MR(x; U) = \operatorname*{argmin}_{x \in X} \max_{u \in U} \max_{y \in X} [u(y) - u(x)]. \tag{2.25}$$

Various applications of decision making with minimax regret criterion have been researched by Boutilier et al. (2001); Boutilier, Das, Kephart, Tesauro, and Walsh (2003a); Boutilier et al. (2003b); Wang and Boutilier (2003); Boutilier et al. (2004c, 2005); Patrascu, Boutilier, Das, Kephart, Tesauro, and Walsh (2005); Braziunas and Boutilier (2007, 2010); Regan and Boutilier (2009).

The minimax criterion does not satisfy the *principle of independence of irrelevant alternatives*. According to this principle, the ranking between two alternatives should be independent of other available alternatives (for example, the violation of this principle could result in a situation where $x \succeq y$ if option z is available, and $y \succeq x$ otherwise). We should note, however, that the principle of independence of irrelevant alternatives is not universally accepted as a prerequisite for rational decision making. However, it is known that *no decision criterion* for strict-uncertainty situations (including the ones discussed in this section) can satisfy all the eight principles of consistent decision making, as stated by French (1986).[1]

Principle of Insufficient Reason This criterion dates back to Pierre Laplace and Jacob Bernoulli, who maintained that complete lack of knowledge about the likelihood of world states should be equivalent to all states having equal probability. Therefore, following this *principle*

[1]The eight axioms of consistent choice are: complete ranking, independence of labeling, independence of value scale, strong domination, independence of irrelevant alternatives, independence of addition of constant to a column (to a choice table), independence of row permutation, and independence of column duplication (French, 1986). The last two axioms encapsulate the concept of strict uncertainty and are violated by the Bayesian decision criteria.

of insufficient reason, an optimal decision maximizes the mean value of possible outcomes:

$$x^* = \arg\max_{x \in X} \mathbb{E}_{u \sim \pi} [u(x)], \tag{2.26}$$

where π is the uniform distribution over U. This criterion is mentioned by Salo and Hämäläinen (2001) as the *central weights* decision rule, and is implicitly employed by Iyengar et al. (2001); Ghosh and Kalagnanam (2003); Toubia et al. (2004), where uncertainty over additive utility functions is characterized by linear constraints on attribute weights. The representative utility function corresponds to the "center" of the weight polytope. Such center could be its actual mass center (i.e., the mean under uniform distribution), or some approximation thereof — a point that minimizes maximal distance to constraint hyperplanes, the center of a bounding ellipsoid, or the average of uniformly sampled points from inside the region.

Acceptability index Finally, there are methods that recommend choosing an alternative based on the set size of supporting utility functions. Lahdelma, Hokkanen, and Salminen (1998) introduce *stochastic multiobjective acceptability analysis* (SMAA), which applies to settings where uncertainty over additive utility functions can be described by linear constraints on the n–1 dimensional weight simplex W. Each alternative is associated with a region of W in which it is optimal. Alternatives are ranked according to *acceptability index*, which is the normalized volume of the weight region in which it is optimal. An alternative with the highest acceptability index is in some sense most likely to be optimal. Like the minimax regret criterion, the acceptability index criterion does not satisfy the principle of independence of irrelevant alternatives.

Various criteria for decision making under strict uncertainty can be grouped into categories based on their general properties. Maximax, maximin, and central values (i.e., optimism-pessimism index) are based on extreme possible values of outcomes. Center-based criteria pick a representative point in the space of feasible utilities. Finally, minimax regret is qualitatively different from the other criteria because it considers pairwise value differences between outcomes.

Unfortunately, different decision rules might prescribe different alternatives. The choice of a decision rule under strict uncertainty should be carefully considered by the decision maker before the elicitation process. French (1986) provides an extensive discussion and critique of various decision criteria under strict uncertainty.

2.2.2.2 Bayesian uncertainty

A true Bayesian would likely reject the very notion of strict uncertainty. An optimal decision is simply the one that maximizes expected value, where expectation is taken with respect to a prior probability distribution π over the set of feasible utilities U:[1]

$$x^* = \arg\max_{x \in X} \mathbb{E}_{u \sim \pi}[u(x)] = \arg\max_{x \in X} EU(x, \pi), \tag{2.27}$$

where $EU(x, \pi)$ is the expected utility of outcome x when π is the probability distribution over utilities (π is also referred to as the elicitor's *belief state* about a subject's preferences). In the case of additional uncertainty over world states, the goal is to maximize *expected* expected utility (Boutilier, 2003).

While most recent work on decision making using distributions over utility functions has been done within the AI community (Chajewska and Koller, 2000; Chajewska et al., 2000; Boutilier, 2002; Braziunas and Boutilier, 2005), the origins of this approach can be traced back to much earlier research in game theory and decision theory. Cyert and de Groot (1979) and de Groot (1983) propose the concept of *adaptive utility*, where a decision maker does not fully know her own utility function until a decision is made. Uncertainty is quantified as a probability distribution over utility function parameters. The distribution is updated by comparing expected utility of an outcome versus its actual utility, which becomes known after the decision is made. Weber (1987) also discusses using expectations over utility functions as a possible criterion for decision making with incomplete preference information. In a related context, probabilistic modeling of possible payoff functions provides the foundation to the

[1] With a prior over utilities, the same decision also minimizes expected regret.

well-established field of Bayesian games (Harsanyi, 1967, 1968).

Boutilier (2003) investigates the conditions under which it is reasonable to model uncertainty over functions measured on the interval scale. By appealing to the foundational axioms of utility theory, it can be shown that the functions are required to be *extremum equivalent*, i.e., they have to share the same best and worst outcomes.

An important issue in the Bayesian approach to modeling uncertainty over utility functions is the choice of prior probability distributions. Ideally, the probability model would be closed under updates (otherwise, it needs to be refit after each response) and flexible enough to model arbitrary prior beliefs. Mixtures of Gaussians (Chajewska et al., 2000), mixtures of truncated Gaussians (Boutilier, 2002), mixtures of uniforms (Boutilier, 2002; Wang and Boutilier, 2003; Braziunas and Boutilier, 2005), and Beta distributions Abbas (2004) are among possibilities proposed in the literature. Priors can also be learned from data — Chajewska et al. (1998) describe a way to cluster utility functions using a database of utilities from a medical domain.

2.2.3 Query selection criteria

A central issue in preference elicitation is the problem of which query to ask at each stage of the process. The value of a query is generally determined by combining the values of possible situations resulting from user responses. As in decision making with incomplete information, query selection is also driven by the ultimate goals of the decision support system. Query selection criteria include fastest reduction of minimax regret or uncertainty, or achieving optimal tradeoff between elicitation costs and predicted improvement in decision quality.

2.2.3.1 Max regret reduction

The minimax regret decision criterion provides bounds on the quality of the decision made under strict uncertainty. When the potential regret associated with each decision is too high, more utility information needs to be elicited. A decision support system can query the user

until the minimax regret reaches some acceptable level, elicitation costs become too high, or some other termination criterion is met.

Each possible response to a utility query results in a new decision situation with a new level of minimax regret (the level of regret cannot increase with more information). The problem is to estimate the value of a query based on the value of possible responses. For example, one could select the query with the best worst-case response, or the query with the maximum average or expected improvement (Wang and Boutilier, 2003; Braziunas and Boutilier, 2010).

Minimax regret reduction queries are also used in the autonomic computing scenario (Boutilier et al., 2003a; Patrascu et al., 2005), eliciting values of non-price features in combinatorial auctions (Boutilier et al., 2004c), and optimizing constrained configurations (Boutilier et al., 2003b, 2005). A more detailed description of these methods is postponed till Section 2.3.6.2.

2.2.3.2 Uncertainty reduction

There is a variety of methods from diverse research areas, such as conjoint analysis and ISMAUT (see Sections 2.3.3 and 2.3.4), whose central idea is to choose queries that reduce the uncertainty over utility functions as much as possible. The set of possible utility functions is commonly represented as a convex polytope in the space of utility function parameters. Each query bisects the polytope by adding a linear constraint. Since the responses are not known beforehand, various heuristics are used to choose the next query. Such heuristics consider the size parity of volumes (Iyengar et al., 2001), as well as their shape (Ghosh and Kalagnanam, 2003; Toubia et al., 2004).

Abbas (2004) proposes an algorithm for query selection in situations where uncertainty over unidimensional utility functions is quantified probabilistically. At each stage, a myopically optimal query provides the largest reduction in the entropy of the joint distribution over utility values. Holloway and White (2003) consider sequentially optimal querying policies for a subclass of problems with additive utility functions and small sets of alternatives. The process is modeled as a special POMDP (see Section 2.3.3 below for a more detailed description).

While such methods strive to minimize the number of queries, they fail to consider the tradeoff between elicitation costs and improvement in decision quality. This is the topic of the next section.

2.2.3.3 Expected value of information

If uncertainty over utilities is quantified probabilistically, the value of a query can be computed by considering the values of updated belief states (one for each possible response), and weighting those values by the probability of corresponding responses. If a sequence of queries can be asked, finding the best elicitation policy is a sequential decision process, providing an optimal tradeoff between query costs (the burden of elicitation) and the value of potentially better decisions due to additional information. However, such a policy is very difficult to compute; therefore, we first describe a myopic approach to choosing the next query.

Myopic EVOI Because of computational complexity of determining full sequential value of a query, it is common to use myopic *expected value of information* (EVOI) to determine appropriate queries (Chajewska et al., 2000). To reduce uncertainty about utility functions, the decision support system can ask questions about the user's preferences. We assume a finite set of available *queries* Q, and, for each query $q \in Q$ — a set of possible user *responses* R_q. Responses to queries depend on the true user utility function u, but might be noisy. A general model that fits many realistic scenarios is a probabilistic *response model* $Pr(r_q|q,u)$, providing the probability of response r_q to the query q when the utility function is u. $Pr(r_q|q,\pi)$ will denote the probability of response r_q with respect to the density π over utility functions:

$$Pr(r_q|q,\pi) = \mathbb{E}_{u\sim\pi}[(Pr(r_q|q,u)]. \tag{2.28}$$

Elicitation of preferences takes time, imposes cognitive burden on users, and might involve considerable computational and financial expense. Such factors can be modeled by assigning each query q a *query cost* c_q.[1] In a Bayesian formulation of the elicitation process, expected

[1] More generally, costs could depend on the true utility function, or be associated with responses.

gains in decision quality should outweigh elicitation costs.

Let's recall that $EU(x, \pi)$ is the expected utility of outcome x when π is the probability distribution over utilities (see Eq. 2.27). Let $MEU(\pi)$ be the *maximum expected utility* of belief state π:

$$MEU(\pi) = \max_{x \in X} EU(x, \pi). \tag{2.29}$$

A response r to a query q provides information about the true utility function and changes our current beliefs from π to π^r according to the Bayes' rule:

$$\pi^r(u) = \pi(u|r) = \frac{Pr(r|u)\pi(u)}{Pr(r|\pi)}. \tag{2.30}$$

Thus, after response r, the maximum expected utility is $MEU(\pi^r)$. To calculate the value of a query, the MEUs of its possible responses should be weighed according to their likelihood. The *expected posterior utility* of the query q is:

$$EPU(q, \pi) = \sum_{r \in R_q} Pr(r|q, \pi)\ MEU(\pi^r). \tag{2.31}$$

The *expected value of information* of the query q is its expected posterior utility minus its current maximum expected utility:

$$EVOI(q, \pi) = EPU(q, \pi) - MEU(\pi). \tag{2.32}$$

EVOI of the query q denotes the gain in expected value of the ultimate decision. A *myopically optimal* query strategy would always select a query whose EVOI is greatest, after accounting for query costs. A *sequentially optimal* strategy would consider the value of future queries when computing the EVOI of the current query. Even though some query might be very costly in short term, it might be able to direct the elicitation process to good regions (in terms of decision quality) of the utility space which might otherwise remain unexplored by the myopic EVOI strategy. The myopic EVOI approach is more popular in practice (used by Chajewska and Koller (2000); Braziunas and Boutilier (2005)) because computational requirements of sequential EVOI are often prohibitive.

Sequential EVOI An obvious way to minimize the shortcomings of myopic querying strategies is to perform a multistage lookahead. Unfortunately, such multistage search would have to be computed online (during the execution of the policy), which might seriously limit its benefits.

Another approach is to compute a sequentially optimal policy offline. Boutilier (2002) introduces the concept of preference elicitation as a POMDP that takes into account the value of future questions when determining the value of the current question. As before, we assume a system that makes decisions on behalf of a user; such a system has a fixed set of choices (actions, recommendations) whose effects are generally known precisely or can be modeled stochastically. The system interacts with a user in a sequential way; at each step it either asks a question, or determines that it has enough information about a user's utility function to make a decision. As each query has associated costs, the model allows the system to construct an optimal interaction policy which takes into account the trade-off between interaction costs and the value of provided information. The approach is discussed in more detail in Section 2.3.6.1.

2.3 Related work on preference elicitation

The last part of the chapter surveys research fields (both historical and from outside computer science) where preference elicitation plays a central role: imprecisely specified multiattribute utility theory (ISMAUT), its extensions to engineering design and configuration problems, conjoint analysis in marketing, and analytical hierarchy process (AHP) in decision analysis. ISMAUT is one of the earlier attempts to consider decision making under partial preference information in classical decision analysis. Conjoint analysis and AHP methods were developed largely in isolation in the fields of marketing research and decision analysis; however, these methods also attempt to solve preference elicitation issues of general interest. We finish by providing an overview of some recent work in preference elicitation in AI.

2.3.1 ISMAUT

One of the earlier attempts to consider decision making under partial preference information is the work on *imprecisely specified multiattribute utility theory*, or ISMAUT, by White et al. (1983, 1984); Anandalingam and White (1993). A similar framework was proposed before by Fishburn (1964) and Sarin (1977). Related research by, e.g., Kirkwood and Sarin (1985); Hazen (1986); Weber (1987), deals with similar issues, even though it is not customarily called ISMAUT.

ISMAUT applies to situations in which the utility function can be written in a normalized additive form, i.e., as a sum of weighted local value functions for each attribute. The decision maker has to choose from a finite set of multiattribute alternatives. The goal of ISMAUT is to restrict the set of alternatives to those that are not dominated by any other alternative, based on the prior information on local value functions, weights (scaling factors), and comparisons between pairs of alternatives. If the reduced alternative set is too big for the decision maker to make a choice, one should assess local value functions or weights more accurately, reduce the set of nondominated alternatives, and continue the process as long as is necessary for optimal alternative selection. An obvious drawback of this scheme is the lack of an intelligent query selection strategy to drive the elicitation process. In the following section, we discuss the research that considers querying strategies in ISMAUT-like elicitation settings.

Let A be the set of available multiattribute alternatives. Each alternative $\mathbf{x} \in A$ is a point in an n-dimensional consequence space: $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. The preference relation over A is representable by an additive utility function:

$$u(\mathbf{x}) = \sum_{i=1}^{n} w_i \, v_i(x_i) = \mathbf{w} \cdot \mathbf{v}(\mathbf{x}),$$

where $\mathbf{w}$ is a vector of weights, and $\mathbf{v}(\mathbf{x})$ is a vector of local value function values of some multiattribute alternative $\mathbf{x}$.

The model can incorporate three types of prior information (or responses to utility queries): comparison of attribute weights w_i, information about local value functions v_i, expressed by

sets of linear inequalities, and pairwise preference statements about alternatives in the set A. In particular,

1) Knowledge about relative importance of the tradeoff weights ("Color is more important than screen size") or bounds on their values ("This attribute's weight is between 0.5 and 1") [1] allows a decision analyst to define a feasible subset $W \subseteq \{\mathbf{w} \in \mathbb{R}^n : w_i \geq 0, \sum_i w_1 = 1\}$ of all possible weights via linear constraints.

2) Similar to statements about weights, ISMAUT incorporates information about individual local value functions by means of linear constraints. If the third attribute is a computer's speed, and the user prefers faster computers, ceteris paribus, then $v_3(\texttt{fast}) \geq v_3(\texttt{slow})$. The user might also be able to provide bounds for local values of specific attribute levels (e.g., $v_3(\texttt{fast}) \in [0.3, 0.7]$). Such linear constraints define the sets $V_1, \ldots, V_n$ of possible local value functions.

3) Finally, even if the user is unable to select the best alternative right away, she might be able to compare some pairs of alternatives. Let J be the set of such comparisons: $J = \{(\mathbf{x}, \mathbf{y}) \in A \times A,\ \mathbf{x} \succeq \mathbf{y}\}$. The set of comparisons J can be used to impose further restrictions on the weight space, because $(\mathbf{x}, \mathbf{y}) \in J$ implies $\mathbf{w} \cdot \mathbf{v}(\mathbf{x}) \geq \mathbf{w} \cdot \mathbf{v}(\mathbf{y})$.

All this prior information defines the set $\mathbf{U}$ of *feasible utility functions* (viz., weights and local value functions). More precisely, the tuple $\langle \mathbf{w}, v_1, \ldots, v_n \rangle \in \mathbf{U}$ if and only if

$$\mathbf{w} \in W,$$

$$v_i \in V_i, \text{ for all } i = 1, \ldots, n,$$

$$\mathbf{w} \cdot [\mathbf{v}(\mathbf{x}) - \mathbf{v}(\mathbf{y})] \geq 0, \text{ for all } (\mathbf{x}, \mathbf{y}) \in J.$$

A decision analyst can use the set of feasible utilities $\mathbf{U}$ to eliminate dominated alternatives. First, let $R(\mathbf{U}) \subseteq A \times A$ be the binary relation of known (inferred) pairwise relationships

[1] Although many authors talk about the "importance" of attributes, we should be aware that weights are nothing more than scaling factors. The statements about weights are nonetheless meaningful: $w_i \geq w_j$ means that outcome $(x_i^\top, \mathbf{x}^\perp_{\{i\}^C})$ is preferred to $(x_j^\top, \mathbf{x}^\perp_{\{j\}^C})$, and $w_i \in [0.5, 1]$ means that $u(x_i^\top, \mathbf{x}^\perp_{\{i\}^C}) \in [0.5, 1]$.

between outcomes:

$$(\mathbf{x}, \mathbf{y}) \in R(\mathbf{U}) \iff \mathbf{w} \cdot [\mathbf{v}(\mathbf{x}) - \mathbf{v}(\mathbf{y})] \geq 0, \text{ for all } \langle \mathbf{w}, v_1, \ldots, v_n \rangle \in \mathbf{U}.$$

This means that the relationship $\mathbf{x} \succeq \mathbf{y}$ is known if and only if $(\mathbf{x}, \mathbf{y}) \in R(\mathbf{U})$. The set of prior stated pairwise preferences J is a subset of $R(\mathbf{U})$.

The set of *nondominated* alternatives $ND(\mathbf{U})$ can be computed using the relation $R(\mathbf{U})$: $\mathbf{x}$ is nondominated if there is no alternative $\mathbf{y}$ such that $(\mathbf{y}, \mathbf{x}) \in R(\mathbf{U})$. Without prior information, $\mathbf{U}$ contains all possible weights and value functions, and $ND(\mathbf{U})$ is equal to the Pareto-optimal set of alternatives. $ND(\mathbf{U})$ is important because the most preferred alternative has to be in it. The goal of ISMAUT is to reduce the set of nondominated alternatives until the user can select the optimal one. More information about the possible local value functions and weights reduces the size of set $\mathbf{U}$, increases the binary relation $R(\mathbf{U})$, and reduces the size of $ND(\mathbf{U})$:

$$\mathbf{U} \subseteq \mathbf{U}' \implies R(\mathbf{U}') \subseteq R(\mathbf{U}) \wedge ND(\mathbf{U}) \subseteq ND(\mathbf{U}').$$

The set $ND(\mathbf{U})$ can be computed from $R(\mathbf{U})$ in polynomial time in the size of the alternative set A, because for each alternative, one needs to check that no other alternative is preferred. The central computational task is therefore to compute $R(\mathbf{U})$ from $\mathbf{U}$. Recall that $(\mathbf{x}, \mathbf{y}) \in R(\mathbf{U})$ if and only if $\mathbf{w} \cdot [\mathbf{v}(\mathbf{x}) - \mathbf{v}(\mathbf{y})] \geq 0$, for all $\langle \mathbf{w}, v_1, \ldots, v_n \rangle \in \mathbf{U}$. This amounts to verifying that

$$\min_{\langle \mathbf{w}, v_1, \ldots, v_n \rangle \in \mathbf{U}} \mathbf{w} \cdot [\mathbf{v}(\mathbf{x}) - \mathbf{v}(\mathbf{y})] \geq 0. \tag{2.33}$$

When both $\mathbf{w}$ and $\mathbf{v}$ are not fully known, this leads to a quadratic program. If only the weights or the value functions are uncertain, the problem is much simpler, and can be solved by a linear program. Appendix A provides more details and references.

Hazen (1986) and Weber (1987) point out that the set of nondominated alternatives is not the same as the set of *potentially optimal* alternatives, which is a subset of $ND(\mathbf{U})$. It is possible that an alternative is not dominated by any other alternative, but is dominated by a collection of alternatives (Weber (1987) calls this *mixed dominance*). Or, alternatively, a

potentially optimal alternative always has a feasible *witness* utility function for which it is an optimal alternative. When local value functions are known, the witness weight vector can be found by solving a linear program.

2.3.2 Engineering design and configuration problems

One field in which applications and extensions of ISMAUT have been proposed is *engineering design*. Design is a multidisciplinary area with no precise definition. Generally, any problem of designing a complex system that has to comply to some performance requirements and satisfy operational constraints can be regarded as an engineering design problem. Examples include communication networks, computer systems, bridges, etc. Some areas within engineering design that have tight connections to AI are *AI in design* (AID), *knowledge-based design systems* (KBDS), and *intelligent computer-aided design* (ICAD) (Brown and Birmingham, 1997). As we shall see, *configuration design* (Wielinga and Schreiber, 1997) is a particularly relevant formalization of the problem with regard to preference elicitation. A recent report by the Board on Manufacturing and Engineering Design (2001) stresses the importance of the decision-theoretic approach in engineering design.

The paper of Sykes and White (1991) on *multiobjective intelligent computer-aided design* (MICAD) extends the ideas of ISMAUT to the problem of configuration design. The design process is viewed as a combination of the progressively acquired *preferential component* and an *a priori operational component*. MICAD thus combines iterative capture of user preferences with the search in the constrained space of feasible designs. Preference elicitation can be directed toward promising (and feasible) regions of the design space, thus avoiding the cost of wasted elicitation effort. On the other hand, the search for optimal designs can be substantially facilitated by preference information.

Intuitively, a configuration problem is that of "configuring" a system. A decision to be taken consists of a number of components, or aspects, which interact in complex ways to produce an outcome. A typical example is configuring a computer system from a set of components

— choosing a processor, compatible memory, peripherals, etc. Possible *configurations* are restricted by hard feasibility constraints. The optimal configuration depends on user preferences; however, those preferences are expressed over *features* or *attributes* of configurations (or *designs*). For example, a user might want a "reliable home computer", which is a point in the *feature space* (or *performance space*), rather than *configuration space*. A mapping from configuration space to feature space induces indirect preferences over configurations.

Let $\mathbf{T} = T_1 \times \ldots \times T_m$ be a multiattribute configuration space, where each T_i is a set of components to choose from. As before, the outcome, or feature, space will be denoted as $\mathbf{X}$. Components represent controllable aspects of a design problem, whereas configuration features allow for a direct expression of user preferences. Feasible configurations $\mathbf{T}^F \subseteq \mathbf{T}$ form a subset of the configuration space; they could be specified by a set of rules, logical formulas, or using *constraint satisfaction problem* (CSP) formulations. A *performance function*[1] $f : \mathbf{T} \mapsto \mathbf{X}$ provides a mapping from configuration space to feature space. The problem is complicated by the fact that this function f might not have any useful mathematical properties (such as continuity, monotonicity or invertibility), and might not be expressed in closed form. Determination of $f(t),\ t \in \mathbf{T}$ might also be costly and require expert analysis or simulation.

The problem of engineering design can thus be summarized follows: given a set of components and features, a set of operational constraints on configurations, a performance function, and a preference relation over the outcome space, find an optimal feasible configuration. This general problem is addressed by Sykes and White (1991); D'Ambrosio and Birmingham (1995) using ISMAUT, and by Boutilier et al. (1997) using CP-nets.

Sykes and White (1991) investigate direct application of ISMAUT ideas to the design process. It is assumed that local value functions are known, so only weights are uncertain. Information about weights can be queried from the user in two ways, already described above: (1) the user can provide direct information about weights, expressed as linear constraints; (2)

[1] Boutilier, Brafman, Geib, and Poole (1997) call the performance function a *causal model*, because it is expressed by a set of logical rules.

the user can compare pairs of designs to induce linear inequalities in the weight space (this requires solving Eq. 2.33). Preference elicitation can occur at any time during an iterative design process. MICAD is presented as a general framework for interactive preference elicitation and search in the space of designs. It is assumed that the search proceeds in stages, at which a finite set of designs is available for the user to evaluate. Two crucial issues are not directly addressed: how to select a set of designs (from potentially exponential number of possibilities) at each stage, and what query selection strategy to follow when eliciting user preferences.

D'Ambrosio and Birmingham (1995) tackle the first issue. They formulate the design engineering problem as a constrained optimization problem. The objective function is an incomplete value function created by pairwise ranking a *random* sample of design alternatives. Operational constraints are modeled as a CSP. Therefore, CSP solution techniques, such as constraint network decomposition and constraint propagation, can be harnessed to facilitate a branch-and-bound search for optimal designs.

2.3.3 Extensions of ISMAUT

Classical ISMAUT is mostly concerned with narrowing the set of alternatives to a manageable size using partial preference information. The following papers address an important issue of how to select queries in a sequential elicitation process. Like in ISMAUT, the additive utility function over attributes is assumed.

The Q-Eval algorithm of Iyengar et al. (2001) asks the user to compare pairs of selected alternatives, and uses the responses to refine the preference model. The local value functions are specified precisely, so utility function uncertainty is represented by linear constraints on the weight space. Given a set of alternatives, the authors address the issue of which pair to present to the user for ranking in a sequential elicitation process. Each response to the query "Is $\hat{\mathbf{x}}$ preferred to $\mathbf{x}$?" adds a linear constraint which reduces the region of feasible weights W. Since a response is not known beforehand, the authors advocate a heuristic of choosing the query that would come closest to bisecting the space of feasible weights. The rationale for this

querying strategy is to shrink the space of possible weights as quickly as possible.

The implementation of Q-Eval employs a number of approximations to ensure practical online performance. First, the number of alternative pairs considered is pruned based on the normal distance of corresponding hyperplanes to the "center" of the region W. Intuitively, hyperplanes close to the center are good candidates for bisecting the region W equally. The notion of center used throughout the paper is that of *prime analytic center*, which is the point that maximizes the sum of log distances to the irredundant hyperplanes defining the region. In case a decision has to be made with uncertain information, the center serves as a representative weight vector. Queries that were not pruned in the previous step are then evaluated based on the volumes of the resulting polytopes (the best query leads to the most equal partition of the weight space). The volumes are approximated by the size of the tightest axis-orthogonal bounding rectangle.

Ghosh and Kalagnanam (2003) consider the same problem and propose to use sampling for determining the center of the weight region W. In particular, they use a hit-and-run sampling technique that employs a Markovian random walk defined on the set W with a uniform stationary distribution. The advocated querying strategy is to ask a query whose corresponding hyperplane is orthogonal to the longest line segment contained in W.

The two query selection methods described above try to minimize the number of queries by shrinking the region of possible weights as fast as possible. However, they do so myopically, without considering the value of a *sequence* of queries. Holloway and White (2003) present a *partially observable Markov decision process* (POMDP) model for sequentially optimal elicitation in an ISMAUT-like setting, where uncertainty about utility functions is specified by linear constraints on weight vectors.

The state space in this POMDP model is the (uncountable) collection of all subsets of tradeoff weights $\{\mathbf{w} \geq 0 : \sum w_i = 1\}$. Intuitively, a system is in state W, if W is the largest region of the weight space constrained by previous elicitation responses. Actions are binary queries asking to compare pairs of alternatives, and observations are yes/no answers to such

queries. It is assumed that there is no noise in user responses. The observation function is the probability of getting a response r to the query q when the true tradeoff weight vector lies in the set W; a uniform probability distribution over the weight space is assumed, although more general probability models could be accommodated. The process moves from one state to another as the feasible weight region shrinks due to linear constraints imposed by responses to queries.

To define the cost structure, the authors use a notion of the solution partition, which divides the set of all weights into convex regions where one alternative dominates all others. Formally, if A is the set of alternatives, a *solution partition* is $\{W_a : a \in A\}$, where

$$W_a = \{\mathbf{w} : \ \mathbf{w} \cdot \mathbf{v}(\mathbf{x}_a) \geq \mathbf{w} \cdot \mathbf{v}(\mathbf{x}_{a'}) \ \forall a' \in A\}. \tag{2.34}$$

Several elicitation goals can be encoded using the cost model. Let T be a maximum number of queries that can be asked (thus, we consider a finite-horizon POMDP). For a given weight vector set W, $c(W, q)$ is the cost of asking the query q, and $\bar{c}(W)$ is the terminal cost of ending up with the set W after all questions have been asked. If the goal is to ask as few queries as possible to determine an optimal alternative, then one can set $\bar{c}(W) = 0$, $c(W, q) = 0$ if there is $a \in A$ such that $W \subseteq W_a$, and $c(W, q) = 1$ otherwise. The optimal policy of this POMDP will ask the fewest queries possible until it finds the smallest $t \leq T$ such that $W^{(t)} \subseteq W_a$ for some $a \in A$. One can similarly define a simple cost function for minimizing the expected uncertainty of knowing the problem's solution after T questions.

Since the POMDP described above is hard to solve in the most general form, the authors concentrate on the case in which restrictions on the cost function guarantee a finite, piecewise linear, representation of the POMDP value function. This is possible if the cost function depends on W only through a finite probability distribution $p^W(\cdot)$ over alternatives. For each $a \in A$, $p^W(a)$ is the probability that alternative a is optimal, given that the true weight vector is in W. Such cost functions are too restrictive for POMDPs that model the optimal tradeoff between elicitation costs and expected improvement in decision quality; however, they can be

used to achieve the two goals mentioned in the previous paragraph.[1]

Holloway and White (2003) do not perform empirical validation of the approach or provide a suitable POMDP solution algorithm; the authors also assume perfect responses to queries. Nevertheless, it is the first attempt to describe a model for sequentially optimal query selection in ISMAUT problems.

2.3.4 Conjoint analysis

Since the original paper by Green and Rao (1971), conjoint analysis has become a major area in marketing research.[2] Conjoint analysis is a set of techniques for measuring consumer tradeoffs among multiattribute products and services. Despite differences in terminology and methodology, conjoint analysis and multiattribute decision analysis (in particular, ISMAUT) deal with similar issues in preference elicitation and modeling.

The goal of conjoint analysis is to decompose consumer preferences over multiattribute *products* (or *profiles*) into component preferences over attributes in order to predict *aggregate* consumer behavior, explain preferences for current products, visualize market segmentation, and help design new products. Thus, the emphasis is generally on predictive and descriptive, rather than prescriptive aspects of consumer behavior.

Usually, an additive utility function is assumed — the total value of a product is the sum of partial contributions (*partworths*) of individual attributes (*features*). Formally, let $y^j = u(\mathbf{x}^j)$ be a specified rating of the product $\mathbf{x}^j$. A general conjoint analysis model is

$$y^j = \sum_i \theta_i \, z_i^j, \tag{2.35}$$

where z_i^j are input variables, y^j is a dependent output variable, and θ_i are parameters to be estimated. Input variables z_i^j depend on the attributes of the product $\mathbf{x}^j$:

[1] For example, the problem of minimizing the number of queries can be encoded by setting $\bar{c}(p^W) = 0$, $c(p^W, q) = 0$ if there exists $a \in A$ such that $p^W(a) = 1$, and $c(p^W, q) = 1$ otherwise.

[2] Green and Srinivasan (1978, 1990) provide key historical surveys of conjoint analysis.

- For continuous attributes whose value is monotonically increasing, $z_i = x_i$. If all attributes are like that, then the model reduces to the familiar linear value function $u(\mathbf{x}) = \sum_i w_i\ x_i$, and parameters θ_i can be viewed as weights w_i.

- For continuous attributes whose local value functions are substantially nonlinear, several z_i variables can be used for approximation. In a case of quadratic function for attribute x_i, two z variables are introduced: one equal to x_i, and the other equal to x_i^2. Such local value models are quite common in conjoint analysis. One example is the *ideal-point model*, where local preference increases quadratically until some ideal-point level, and decreases after that.

- For discrete binary attributes with two levels $x_i^\top$ and $x_i^\bot$, $z_i = 1$ if $X_i = x_i^\top$, and $z_i = 0$ if $X_i = x_i^\bot$. Then an estimated parameter θ_i can be thought of as a local value of the best level of x_i.

- Discrete attributes with K levels are converted into $K - 1$ binary "dummy" attributes. Constraints on indicator variables z_i are added to ensure consistency of the 1-of-K representation.

Given preference information about whole products (such as ordinal or cardinal product ranking, comparison, or preferred choice from a set of products), some form of regression is used to find parameters that are *most consistent* with specified preferences, which are usually aggregate. For example, a common type of application is to elicit preferences over full profiles using a rating or ranking scale, and then estimate attribute partworths by least-squares regression. The underlying assumption is that ranking or rating full products is easier than providing attribute partworths, as long as the number of attributes is small.

Many aspects of preference elicitation considered in this thesis have their equivalents in conjoint analysis, too. Approaches are differentiated according to data collection formats (i.e., "query types"), question design ("query selection"), and parameter estimation procedures ("decision making with incomplete information"). The most common data collection format is full

profile evaluation, where a user is asked to order all products (*stimuli*) in a given set, or provide a metric rating of each stimulus. Of course, the user's burden grows dramatically with the size of stimulus set. Some methods therefore employ partial profile evaluations. *Choice-based conjoint analysis* (CBC) is a popular compromise technique, where instead of ranking all profiles, a user is asked to choose the most preferred from a given a set. *Metric paired-comparison* format asks to consider only pairs of profiles, but expects quantitative answers regarding relative preference.[1]

Until recently, most applications of conjoint analysis either presented the same questions to all respondents, blocked them across sets of respondents, chose randomly, or adapted them based on responses from prior respondents. Adaptive question design for individual respondents in the manner of ISMAUT was first considered by Toubia et al. (2003, 2004) in metric paired-comparison and CBC settings. This new approach, termed the *polyhedral method*, works by iteratively constraining the polyhedron of feasible subutility (partworth) values. The attributes are discrete and binary (multilevel attributes can be represented using dummy variables), so each product is represented by a point in the space of attribute partworths. In CBC, binary comparison questions result in a separating hyperplane that cuts the polyhedron of feasible subutilities. More generally, a respondent is presented with a set of products, and asked to choose one of them. A choice set of size k defines $k(k-1)/2$ possible hyperplanes; for each of k choices available, the $k-1$ *new* hyperplanes induced by that choice determine the new polyhedron.

In polyhedral methods, the goal is to reduce the size of uncertainty polyhedron as fast as possible. Questions are designed to partition the polyhedron into approximately equal parts; in addition, shape heuristics are used to favor cuts that are perpendicular to long axes. Since the problem is computationally hard, many approximations similar to Q-Eval (Iyengar et al., 2001) are employed. The polyhedron's volume is approximated by a bounding ellipsoid, and its

[1] In practice, a user is usually provided with a set of qualitative choices specifying by how much product x is preferable to product y (e.g., "I like x much more than y", "I like x a little more than y", "I like x as much as y", etc.); these choices are then converted to a quantitative scale.

center by the analytic center. Then, k points at which $k/2$ longest axes intersect the polyhedron are used to select k profiles for the next choice-based query. This technique is extended to metric paired-comparison queries in Toubia et al. (2003).

Conjoint analysis and decision analysis have largely developed in parallel, without much interaction. However, recent emphasis on sequential preference elicitation in both fields presents opportunities for fertile interaction. Conjoint analysis offers query formats that have been validated in practice, and experimental domains in consumer research. Its limitations include reliance on full profile queries,[1] which work only for products with a few (usually less than ten) attributes, common assumptions of attribute independence, and lack of well-defined explicit elicitation optimization criteria.

2.3.5 Analytic hierarchy process

Analytic hierarchy process (AHP) is an alternative method of decision analysis developed by Saaty (1977, 1980). The main ideas of the AHP method can be explained in comparison to additive value theory (French, 1986), although the connection between the two approaches was developed well after the original work on AHP.

The problem is to select the best alternative from the set of D multiattribute alternatives $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^D$ under certainty. Each alternative is measured against N attributes: $\mathbf{x}^k = (x_1^k, x_2^k, \ldots, x_N^k)$. The value function is represented as a weighted sum of strictly positive local value functions v_i:

$$v(\mathbf{x}^k) = \sum_{i=1}^{N} w_i v_i(x_i^k) = \sum_{i=1}^{N} w_i v_i^k, \tag{2.36}$$

where v_i^k is the local value of the kth alternative on the ith attribute. The weights and local value functions are not normalized to $[0, 1]$.

The main difference between AHP and classical decision analysis lies in the elicitation of weights and local value functions. Instead of direct responses regarding attribute weights and

[1] There are methods of conjoint analysis that do not employ full product comparisons, but they are less popular and not as well grounded theoretically from the decision theory perspective.

local value functions, AHP assumes that a user can instead provide all the entries of the so-called *positive reciprocal matrices*. For each attribute a, the entries can be thought of as ratios of local value functions:

$$R_a = \begin{pmatrix} 1 & r_a^{12} & \dots & r_a^{1D} \\ 1/r_a^{12} & 1 & \dots & r_a^{2D} \\ \vdots & \vdots & \dots & \vdots \\ 1/r_a^{1D} & 1/r_a^{2D} & \dots & 1 \end{pmatrix} = \begin{pmatrix} v_a^1/v_a^1 & v_a^1/v_a^2 & \dots & v_a^1/v_a^D \\ v_a^2/v_a^1 & v_a^2/v_a^2 & \dots & v_a^2/v_a^D \\ \vdots & \vdots & \dots & \vdots \\ v_a^D/v_a^1 & v_a^D/v_a^2 & \dots & v_a^D/v_a^D \end{pmatrix}, \tag{2.37}$$

where r_a^{ij} is the *ratio* of local values v_a^i and v_a^j. Besides N attribute matrices, an additional matrix R is elicited to provide information about relative importance of attributes.

$$R = \begin{pmatrix} w_1/w_1 & w_1/w_2 & \dots & w_1/w_N \\ w_2/w_1 & w_2/w_2 & \dots & w_2/w_N \\ \vdots & \vdots & \dots & \vdots \\ w_N/w_1 & w_N/w_2 & \dots & w_N/w_N \end{pmatrix}. \tag{2.38}$$

The entries of R can be interpreted as ratios of attribute weights.

Given the entries of the positive reciprocal matrices, AHP derives the weights and local value functions for the attributes. If the matrix entries were consistent, such a derivation would amount to solving a simple system of linear equations. In the likely case of inconsistent entries, the local value functions and weights are estimated using one of several averaging techniques (eigenvector-based estimation is commonly advocated). The alternatives are ultimately ranked by the resulting additive value function.

The key issue is eliciting the positive reciprocal matrices R_a. For attribute matrices, the user is asked to compare pairs of alternatives on each attribute.[1] For a pair of alternatives $\mathbf{x}^i$ and $\mathbf{x}^j$ compared on attribute a, the ratio r_a^{ij} is 1, if $\mathbf{x}^i$ is *equally* preferred to $\mathbf{x}^j$, 3 — *weakly* preferred, 5 — *strongly* preferred, 7 — *demonstrably* preferred, and 9 — *absolutely* preferred.

[1] *Ceteris paribus* with respect to remaining attribute values should certainly be assumed, although such issues, and many other, are often skirted in AHP literature.

The weight matrix is elicited by a similar process — one attribute can be "equally important", "weakly more important", "strongly more important", "demonstrably more important", and "absolutely more important" than another.

Issues such as elicitation costs, decision making with incomplete information, and query selection criteria are as important in the AHP as in classical decision theory. Salo and Hämäläinen (1995, 2001, 2004) maintain that decisions should be made with incomplete information if elicitation costs outweigh potential improvement in decision quality. Preference uncertainty is described by bounds on value function ratios (i.e., the entries of positive reciprocal matrices). Several decision criteria are discussed, and "central values" approach (see Section 2.2.2.1) is favored on the grounds of empirical simulations. Query selection is not addressed.

AHP is a controversial method (see, e.g., the critiques by French (1986); Salo and Hämäläinen (1997)). While quite popular in practice,[1] it is not as well grounded theoretically as classical decision theory. One problem is that while local value functions are interval scales, the construction of positive reciprocal matrices assumes that they are *ratio* scales;[2] AHP fails to provide an axiomatic basis for such a strong assumption. Elicitation of matrix entries is also problematic, since it is hard to provide an exact semantic meaning to the AHP queries. The nine-point scale is a source of further controversy. If level 1 of an attribute is absolutely preferred to level 2, and level 2 is absolutely preferred to level 3, then the ratio of level 1 and level 3 should be $9 \times 9 = 81$. However, the scale allows only numbers from 1 to 9. Finally, AHP violates the principle of independence of irrelevant alternatives (i.e., the principle that ranking between two alternatives should be independent of other available alternatives).

[1] Among the main reasons for AHP popularity is the relative simplicity of elicitation queries: to obtain a total ranking of multiattribute alternatives, a user is only asked to provide a qualitative comparison between *pairs* of attributes.

[2] Functions on a ratio scale are unique up to positive scaling.

2.3.6 Preference elicitation in AI

2.3.6.1 Bayesian approach

If uncertainty about utility functions can be quantified probabilistically, then one can design preference elicitation strategies that optimally balance the tradeoff between elicitation effort and the impact of information on the decision quality. Until recently, this approach has been explored very little. In this section, we take a look at some of the attempts to solve this problem by AI researchers. In Chapter 4, we describe our own contributions to the Bayesian elicitation of GAI utilities.

Myopic EVOI Chajewska et al. (2000) were arguably the first to adopt a consistent Bayesian view of the preference elicitation problem. If the utility function is not fully known, it is treated as a random variable drawn from the prior distribution (Chajewska and Koller, 2000). The value of a decision in an uncertain situation is computed by taking an expectation over all possible utility functions. Furthermore, the value of a query is simply its expected value of information.

The proposed framework leads to a simple elicitation algorithm. At each step, the query with the highest EVOI is asked, and the distribution over utilities is updated based on user responses. The process stops when the expected value of a decision meets some termination criteria. Because the sequential EVOI (which takes into consideration all possible future questions and answers) is hard to compute, the value of a query is approximated by the *myopic* EVOI (see Eq. 2.32).

In the prenatal diagnosis decision model described in the paper, the outcome space of size D is discrete and unstructured (flat). Therefore, the space of all utility functions can be represented by a D-dimensional unit hypercube. A multivariate Gaussian distribution (restricted to [0,1]) is used to model the prior over utilities. After a binary standard gamble query ("Is utility of outcome x greater than p?"), the resulting posterior becomes a truncated Gaussian, which is then *approximated* by a new multivariate Gaussian distribution. Experimental results on the

domain with 108 outcomes show that very few queries are needed to reduce the expected utility loss below a small threshold.

Preference elicitation as a POMDP To overcome the shortcomings of myopic EVOI approaches, the preference elicitation problem can be modeled as a POMDP (Boutilier, 2002). The state space of the preference elicitation POMDP is the set of possible utility functions U; actions can be either queries about a user's utility function Q or terminal decisions; observation space is the set of possible responses to queries R. The dynamics of the system is simplified by the fact that the state transition function is trivial: the underlying utility functions never change throughout the interaction process; the observation function is the response model which maintains a probability distribution of a particular response to a given query for a specific utility function; and, the reward function simply assigns costs to queries and expected utilities to final decisions.

Solving the preference elicitation POMDP is a difficult task. In realistic situations, the state space is continuous and multi-dimensional, so standard methods for solving finite-state POMDPs are no longer applicable. Boutilier (2002) presents a value-iteration based method that exploits the special structure inherent in the preference elicitation process to deal with parameterized belief states over the continuous state space; belief states are represented by truncated Gaussian or uniform mixture models. With standard gamble comparison queries that "slice" the density vertically ("Is utility of outcome x greater than p?), updated distributions remain conjugate to the prior. The POMDP is solved by approximating the value function using asynchronous value iteration.

The preference POMDP can also be solved using policy-based methods. Braziunas and Boutilier (2004) describe an algorithm BBSLS that performs stochastic local search in the space of finite state policy controllers. In the case of continuous utility functions, it is possible to *sample* a number of states (utility functions) at each step, and calculate the observation and reward functions for the sampled states. The results for a very small preference elicitation prob-

lem (Boutilier, 2002) provide the proof-of-concept verification of the policy-based approach. There is a lot of room for future research in this area as POMDP-based methods so far can only solve unrealistically small problems.

2.3.6.2 Minimax regret approach

Minimax regret criterion can be used both for making robust decisions under strict uncertainty and for driving an elicitation process. Contrary to the methods in the previous section, the quality (difference from optimal) of a minimax regret optimal decision can be bounded; these bounds can be tightened with further elicitation effort. Minimax regret methods have been applied to several areas of AI, including auctions (Wang and Boutilier, 2003), autonomic computing (Boutilier et al., 2003a; Patrascu et al., 2005), combinatorial auctions (Boutilier et al., 2004c), and constrained configuration problems (Boutilier et al., 2003b, 2005). In Chapters 5 and 6, we describe further extensions of minimax regret based approaches to GAI utilities.

Wang and Boutilier (2003) consider a simple problem with a flat outcome space and binary standard gamble queries. A response to a query results in a new decision situation with a new level of minimax regret. The (myopic) value of a query is a function of response values. The authors consider three ways of combining response values: maximin improvement (select the query with the best worst-case response), average improvement (select the query with the maximum average improvement), and expected improvement (select the best query based on improvements weighted by the likelihood of responses). It turns out that the expected improvement criterion, combining a Bayesian query selection strategy and a robust minimax regret decision criterion, performs best experimentally and is not subject to stalling — the situation when no query improves the minimax regret level. Using binary standard gamble queries, the querying strategy can be optimized analytically.

Boutilier et al. (2003b) address the problem of choosing the best configuration from the set of feasible configurations encoded by hard constraints. It is assumed that preferences over configurations can be represented by a GAI utility function; however, this function is imprecisely specified by bounds on GAI subutility function values. The authors propose the use of minimax regret as a suitable decision criterion and investigate several algorithms based on mixed integer linear programming to compute regret-optimizing solutions efficiently.

Boutilier et al. (2005) concentrate on the utility elicitation aspect and provide an empirical comparison of minimax regret reduction strategies in GAI utility models, where uncertainty over utilities is expressed by bounds on local factor values. The objective is to refine utility uncertainty and reduce minimax regret with as few queries as possible. The queries are *bound* queries: the user is asked whether a specific local utility parameter lies above a certain value. A positive response raises the lower bound, while a negative response lowers the upper bound of a local subutility value.

The *halve largest gap* (HLG) elicitation strategy recommends a query at the midpoint of the bound interval of the GAI factor setting with the largest gap between upper and lower bounds. HLG uniformly reduces uncertainty over the entire utility space and therefore provides the best theoretical minimax regret reduction guarantees. It is related to polyhedral methods (with rectangular polytopes) in conjoint analysis which attempt to maximally reduce uncertainty with each query. Another, *current solution* (CS), strategy, uses heuristics to focus on *relevant* parts of the utility space and works better in practice. CS relies on two special outcomes that are directly involved in calculating the regret level: $\mathbf{x}^*$, the minimax optimal configuration, and $\mathbf{x}^w$, the *witness* point that maximizes the regret of $\mathbf{x}^*$. The CS strategy considers only local factor settings that are part of these two special outcomes and asks about the one with the largest gap. A few other heuristic strategies are also tested in experiments.

The minimax regret criterion can also be applied to a completely different domain of autonomic computing (Boutilier et al., 2003a; Patrascu et al., 2005). To solve the problem of

optimal resource allocation one needs to know the utility of different levels of resource applied to the distributed computing elements. However, even a single evaluation of the utility function is very costly. Patrascu et al. (2005) investigate how to sample a monotonic non-decreasing utility function with a continuous unidimensional domain using strategies similar to CS and HLG.

Chapter 3

GAI utility

Contents

This chapter is about the decision-theoretic foundations that support *local* elicitation of GAI utilities. It is based on the work that first appeared in the paper by Braziunas and Boutilier (2005).

The first section introduces GAI utility models. The second section deals with semantically sound representation of *local* structure in GAI utilities; this representation serves as the primary model of preferences in the remaining chapters of the thesis. Local structure facilitates not only representation, but also *elicitation* of utility parameters. The third section of the chapter enumerates the set of queries for elicitation of GAI utility parameters that are used in our elicitation framework (in both Bayesian and strict uncertainty settings). We discuss related work and summarize our contribution in the concluding section.

3.1 GAI utility model

This section introduces formal definitions of the generalized additive independence and Fishburn's GAI representation theorem (Fishburn, 1967b, 1970). After introducing the key concepts of *reference* and *basic* outcomes, we prove the GAI structure lemma, which is then used to outline the proof of the GAI representation theorem. The concepts and insights gained in this section will be applied when investigating the GAI local structure in the following sections.

3.1.1 Generalized additive independence

Following the notation introduced in Chapter 2, we assume a set of N attributes $X_1, X_2, \ldots, X_N$, each with finite domains. All the possible instantiations of attribute values define a set of *out-*

comes $\mathbf{X} = X_1 \times \cdots \times X_N$. We also consider a collection of M attribute subsets, or *factors*, that *cover* the set of all attributes: $F_1 \cup F_2 \cdots \cup F_M = \{X_1, X_2, \ldots, X_N\}$. A factor $F_I = \{X_i \mid i \in I\}$ contains the attributes whose indices are in the index set $I \subseteq \{1, \ldots, N\}$. The factors (and their associated sets of indices) will be commonly enumerated from 1 to M: $F_1, F_2, \ldots, F_M$. In such a case, it will be assumed that $F_j = F_{I_j}$.

Given an index set $I \subseteq \{1, \ldots, N\}$, we define $\mathbf{X}_I = \times_{i \in I} X_i$ to be the set of *partial outcomes* (or *suboutcomes*) restricted to attributes indexed by I. For a factor F_j, $\mathbf{x}_{I_j}$, or simply $\mathbf{x}_j$, is a particular instantiation of its attributes. Let $\mathcal{P}$ be the set of all lotteries (probability distributions) on $\mathbf{X}$, and $\mathcal{P}_I$ be the set of all lotteries on $\mathbf{X}_I$. For $P \in \mathcal{P}$, P_I is the marginal lottery of P over $\mathbf{X}_I$. By convention, $P_j = P_{I_j}$ will denote the marginal lottery over all the partial outcomes in factor F_j.

Definition 3.1 (Generalized additive independence condition) Fishburn (1967b)
Factors $F_1, \ldots, F_M$ are *(generalized) additively independent* if

$$(P_1, \ldots, P_M) = (Q_1, \ldots, Q_M) \implies P \sim Q, \text{ for all } P, Q \in \mathcal{P},$$

where $P_j, Q_j \in \mathcal{P}_j$ are the j^{th} marginals of $P, Q \in \mathcal{P}$ (for $j = 1, \ldots, M$). That is, the GAI condition holds if and only if a decision maker is indifferent between two lotteries whenever their marginal distributions on $\mathbf{X}_1, \ldots, \mathbf{X}_M$ are the same.

Example 3.1 In the simple two-attribute example by Bacchus and Grove (1995), the outcome space is defined by binary $health$ (with values H and $\bar{H}$) and $wealth$ (with values W and $\bar{W}$) attributes. The utilities are as follows: $u(HW) = 5$, $u(H\bar{W}) = 2$, $u(\bar{H}W) = 1$, $u(\bar{H}\bar{W}) = 0$. If the two attributes (which can be viewed as factors that contain only one attribute each) were (generalized) additively independent, then a decision maker would have to be indifferent between any two lotteries that have the same marginals on each attribute. Let $P = \langle 0.25, HW; 0.25, H\bar{W}; 0.25, \bar{H}W; 0.25, \bar{H}\bar{W}\rangle$, and $Q = \langle 0.5, HW; 0, H\bar{W}; 0, \bar{H}W; 0.5, \bar{H}\bar{W}\rangle$. Both P and Q have the same marginals on $health$ and $wealth$ attributes: $P_{health} = Q_{health} = \langle 0.5, H; 0.5\bar{H}\rangle$, and $P_{wealth} = Q_{wealth} = \langle 0.5, W; 0.5\bar{W}\rangle$. However, the decision maker is not

indifferent between P and Q (since $u(P) = 2$ and $u(Q) = 2.5$), and therefore, the two factors (attributes) in this example are not additively independent. Intuitively, the decision maker prefers lotteries in which health and wealth are positively correlated. *Generalized* additive independence covers more general cases where factors contain several attributes, and are not necessarily disjoint.

The term *generalized additive independence* was popularized by Bacchus and Grove (1995), and is currently widely accepted in the AI literature. Originally, Fishburn (1967b) used the term *interdependent additivity* to denote this concept.

3.1.2 GAI representation theorem

As with additive utilities, the GAI preferential independence condition has a simple numerical representation in a utility function whose structure reflects factor independence.

Theorem 3.1 (GAI representation) Fishburn (1967b)

Let u be a utility function representing user preferences over $\mathbf{X}$. The GAI condition holds if and only if there exist functions $u_1, \ldots, u_M$ on $\mathcal{P}_1, \ldots, \mathcal{P}_M$ such that

$$u(P) = \sum_{j=1}^{M} u_j(P_j). \tag{3.1}$$

By the expected utility theorem, $u(P) = \sum_{\mathbf{x}} P(\mathbf{x})u(\mathbf{x})$, and $u_j(P_j) = \sum_{\mathbf{x}_j} P_j(\mathbf{x}_j)u_j(\mathbf{x}_j)$.

We will consider the proof of this theorem below in Section 3.1.5.

By holding an outcome $\mathbf{x} \in \mathbf{X} \subset \mathcal{P}$ to be a degenerate lottery in which $\mathbf{x}$ occurs with certainty, we can rewrite the above GAI equation as:

$$u(\mathbf{x}) = u_1(\mathbf{x}_{I_1}) + \ldots + u_M(\mathbf{x}_{I_M}). \tag{3.2}$$

The functions $u_1, \ldots, u_M$ on $\mathbf{X}_{I_1}, \ldots, \mathbf{X}_{I_M}$ will be referred to as *subutility* functions.

Example 3.2 Let's consider the simplest non-trivial GAI model with three attributes X_1, X_2 and X_3 grouped into two GAI factors $I_1 = \{1, 2\}$, and $I_2 = \{2, 3\}$. In a travel planning

scenario described in the introduction, X_1 could be the airline, X_2 the flight class, and X_3 the flight length. If, quite realistically, the strength of user preferences over flight class (e.g., economy or business) and airline depends on the flight length, then the simple additive user utility function

$$u_A(\mathbf{x}) = u_1(x_1) + u_2(x_2) + u_3(x_3)$$

cannot adequately represent her true preferences. In contrast, a GAI function with two factors is much more expressive:

$$u_{GAI}(\mathbf{x}) = u_1(x_1, x_2) + u_2(x_2, x_3).$$

In the following subsections, we will sketch the proof of the GAI representation theorem (Fishburn, 1967b, 1970). This will allow us to better understand the notations and concepts that are used later to describe the local structure of the GAI model. But first we define the notions of *reference* and *basic* outcomes (the terminology is ours).

3.1.3 Reference outcome and basic outcomes

By decomposing the set of attributes into factors, GAI models impose a certain structure on the space of all outcomes $\mathbf{X}$. This structure is anchored by the *reference* outcome. In general, the reference outcome can be any arbitrary fixed outcome in $\mathbf{X}$. In elicitation, it is used as a stable reference point when comparing and evaluating other outcome utilities. In most cases, the reference outcome will be denoted as $\mathbf{x}^0 = (x_1^0, x_2^0, \ldots, x_N^0)$.

By using a reference outcome, any outcome $\mathbf{x} \in \mathbf{X}$ can be "projected" to a "subset space" indexed by I, resulting in the outcome $\mathbf{x}[I]$. For any $\mathbf{x} \in \mathbf{X}$, $\mathbf{x}[I]$ is an outcome where attributes of $\mathbf{x}$ not indexed by I are "clamped" at the reference values: $X_i = x_i$ if $i \in I$, and $X_i = x_i^0$ if $i \notin I$. Thus, if $\mathbf{x} = (x_1, x_2)$, then $\mathbf{x}[\{1\}] = (x_1, x_2^0)$, $\mathbf{x}[\{1, 2\}] = (x_1, x_2) = \mathbf{x}$, and $\mathbf{x}[\emptyset] = (x_1^0, x_2^0) = \mathbf{x}^0$.

We also use similar notation for projections of *partial* outcomes. If $\mathbf{x}_j$ is a local configuration of factor F_j, then $\mathbf{x}_j[I]$ is a local configuration of factor F_j in which $X_i = x_i$ if $i \in I \cap I_j$,

$\mathbf{X}_1$	
$\mathbf{x}_1^1 =$	$\underline{x}^1 y^1$
$\mathbf{x}_1^2 =$	$x^2 y^1$
$\mathbf{x}_1^3 =$	$\underline{x^1 y^2}$
$\mathbf{x}_1^4 =$	$x^2 \underline{y}^2$

(a) Factor F_1

$\mathbf{X}_2$	
$\mathbf{x}_2^1 =$	$y^1 \underline{z}^1$
$\mathbf{x}_2^1 =$	$\underline{y^2 z^1}$
$\mathbf{x}_2^1 =$	$y^1 z^2$
$\mathbf{x}_2^1 =$	$\underline{y}^2 z^2$

(b) Factor F_2

$\mathbf{X}$	
$\underline{x}^1 y^1 \underline{z}^1 =$	$\mathbf{b}^{1,1} = \mathbf{b}^{2,1}$
$x^2 y^1 \underline{z}^1 =$	$\mathbf{b}^{1,2}$
$\underline{x^1 y^2 z^1} =$	$\mathbf{b}^{1,3} = \mathbf{b}^{2,2} = \mathbf{x}^0$
$x^2 \underline{y^2 z^1} =$	$\mathbf{b}^{1,4}$
$\underline{x}^1 y^1 z^2 =$	$\mathbf{b}^{2,3}$
$x^2 y^1 z^2$	
$\underline{x}^1 y^2 z^2 =$	$\mathbf{b}^{2,4}$
$x^2 \underline{y}^2 z^2$	

(c) Basic outcomes

Table 3.1: An illustrative example of a GAI model with three binary attributes x, y, z and two factors $F_1 = \{X, Y\}$ and $F_2 = \{Y, Z\}$. The reference outcome is $\mathbf{x}^0 = \underline{x^1 y^2 z^1}$; in all configurations shown above, the reference values are underlined. Table (a) enumerates all the local configurations of factor F_1, and Table (b) enumerates all the local configurations of factor F_2. Table (c) shows all outcomes in $\mathbf{X}$, and specifies which of those outcomes are basic outcomes. For example, outcome $x^2 y^2 z^2$ is not a basic outcome for any factor, whereas outcome $x^2 \underline{y^2 z^1} = \mathbf{b}^{1,4}$ is the fourth basic outcome of factor F_1 (because $\mathbf{x}_1^4 = x^2 \underline{y}^2$ is the fourth local configuration in factor F_1, and $\underline{z}^1$ is the remaining attribute *outside* the factor, fixed at the reference level). Some outcome might also be a basic outcome for *several* different factors: $\underline{x}^1 y^1 \underline{z}^1 = \mathbf{b}^{1,1} = \mathbf{b}^{2,1}$ is a basic outcome for factors F_1 and F_2.

and $X_i = x_i^0$ if $i \in I^C \cap I_j$ (we use intersection with I_j to ensure that we deal only with attributes within the factor F_j).

When the subset I in the reference projection $\mathbf{x}[I]$ is the index set of some GAI factor F_I, $\mathbf{x}[I]$ is called a *basic outcome* for factor F_I. Given GAI factors $F_1, \ldots, F_M$, a basic outcome for factor F_j is any outcome whose attributes outside the factor F_j are fixed at reference levels.

Definition 3.2 An outcome $\mathbf{b} \in \mathbf{X}$ is a *basic outcome for factor* F_j if $\mathbf{b} = \mathbf{x}[I_j]$ for some $\mathbf{x} \in \mathbf{X}$.

If a factor F_j has N_j local configurations (instantiations of factor attributes), it also has N_j corresponding basic outcomes. The k^{th} basic outcome for factor F_j will be denoted $\mathbf{b}^{j,k}$. In Table 3.1(c), the basic outcomes are listed in the second column.

Observation 3.1 *If $I \subseteq I_j$, then $\mathbf{x}[I]$ is a basic outcome for factor F_j.*

Proof In $\mathbf{x}[I]$, all attributes indexed by I^C are fixed at reference values. Since $I_j^C \subseteq I^C$ (because $I \subseteq I_j$), all attributes indexed by I_j^C (which is a subset of I^C) are fixed at reference values. Therefore, $\mathbf{x}[I]$ is a basic outcome for factor F_j. □

As an example, consider an arbitrary basic outcome $\mathbf{x}_{I_j}\mathbf{x}^0_{I_j^C}$ for factor F_j, where the factor configuration is $\mathbf{x}_{I_j}$, and all the other attributes are fixed to the reference level $\mathbf{x}^0_{I_j^C}$. Then, this observation simply states that $\mathbf{x}_{I_j}\mathbf{x}^0_{I_j^C}$ is still a basic outcome for factor F_j if some of the values in $\mathbf{x}_{I_j}$ happen to be reference values. Yet another way of looking at this is to realize that if $I \subseteq I_j$, then applying the reference projection does not change the result: $(\mathbf{x}[I])[I_j] = \mathbf{x}[I]$, which, by definition, means that $\mathbf{x}[I]$ is a basic outcome for F_j. It also follows that the reference outcome $\mathbf{x}^0 = \mathbf{x}[\emptyset]$ is a basic outcome for any factor (since $\emptyset \subset I_j$ for all j).

3.1.4 GAI structure lemma

We are now ready to introduce the *GAI structure lemma* (our terminology), that captures all dependencies intrinsic to GAI utility functions and serves as a semantic foundation for GAI models. In the following subsection, this lemma is used in the proof of the GAI representation theorem.

Lemma 3.1 (GAI structure lemma) Fishburn (1967b)

Let u be a utility function representing user preferences over $\mathbf{X}$. If the GAI condition holds, then for all $\mathbf{x} \in \mathbf{X}$:

$$u(\mathbf{x}) = \sum_{j=1}^{M} (-1)^{j+1} \sum_{\substack{S \subseteq \{1,\ldots,M\}, \\ |S|=j}} u(\mathbf{x}[\cap_{s \in S} I_s]). \tag{3.3}$$

Assuming the GAI condition, Lemma 3.1 provides a way to write the utility of any outcome $\mathbf{x}$ as a sum of utilities of certain other *basic* outcomes. As defined above, these outcomes are related to $\mathbf{x}$ in a specific way: in each of them, some attributes are set to the same levels as in outcome $\mathbf{x}$, while remaining attributes are at their reference values. We consider certain properties of GAI models before proving Lemma 3.1.

Observation 3.2 *All outcomes on the right side of the Equation 3.3 are basic outcomes.*

Proof All outcomes on the right side have the form $\mathbf{x}[\cap_{s \in S} I_s]$, where I_s is an index set of the factor F_s. Since $\cap_{s \in S} I_s \subseteq I_{s'}$ for any $s' \in S$, by Observation 3.1, $\mathbf{x}[\cap_{s \in S} I_s]$ is a basic outcome for factor $F_{s'}$ (in fact, it is a basic outcome for all factors indexed by elements of S). □

Going back to the Eq. 3.3, we see that it is a variant of the *inclusion-exclusion* principle applied to factored utilities. On the right side of the equation, we add utilities of outcomes that involve intersections of sets of factor attribute indices of size one, subtract utilities of outcomes with intersections of sets of factor attribute indices of size two, and so on, until we reach M, the number of factors in the GAI model.

Example 3.3 (Lemma 3.1) Given two factors F_1 and F_2, we have:

$$u(\mathbf{x}) = u(\mathbf{x}[I_1]) + u(\mathbf{x}[I_2]) - u(\mathbf{x}([I_1 \cap I_2]).$$

If $I_1 = \{1, 2\}$, and $I_2 = \{2, 3\}$,

$$u(x_1, x_2, x_3) = u(x_1, x_2, x_3^0) + u(x_1^0, x_2, x_3) - u(x_1^0, x_2, x_3^0).$$

Applying the GAI structure lemma to outcomes from Table 3.1(c), we see that the utility of a non-basic outcome $x^2y^2z^2$ can always be written in terms of utilities of three basic outcomes:

$$u(x^2y^2z^2) = u(x^2y^2z^1) + u(x^1y^2z^2) - u(x^1y^2z^1).$$

With three factors F_1, F_2, F_3, we have:

$$\begin{aligned} u(\mathbf{x}) = {} & u(\mathbf{x}[I_1]) + u(\mathbf{x}[I_2]) + u(\mathbf{x}[I_3]) \\ & - u(\mathbf{x}[I_1 \cap I_2]) - u(\mathbf{x}[I_1 \cap I_3]) - u(\mathbf{x}[I_2 \cap I_3]) \\ & + u(\mathbf{x}([I_1 \cap I_2 \cap I_3]). \end{aligned}$$

Probability distributions P and Q

To prove Lemma 3.1, we first introduce two special probability distributions P and Q.

Let R denote the power set of factor indices $\{1, \ldots, M\}$; also, let R_e be the set of all even-size subsets in R, and R_o be the set of all odd-size subsets.

Observation 3.3 *R_e contains the same number of elements as R_o. Since the size of R is 2^M,*

$$|R_e| = |R_o| = 2^{M-1}.$$

Proof To see this, consider a binomial expansion of $(1 + (-1))^M$:

$$0 = (1-1)^M = \binom{M}{0} - \binom{M}{1} + \binom{M}{2} - \cdots + (-1)^j \binom{M}{j} + \cdots + (-1)^M \binom{M}{M}.$$

Since all the elements with negative coefficients represent the counts of odd-size subsets, and all the elements with positive coefficients represent the counts of even-size subsets in R, the number of odd-size subsets is equal to the number of even-size subsets. □

We also define $R_j \subseteq R$ (for $j = 0, \ldots, M$) to be the set of subsets in R whose size is j. The size of R_j is $\binom{M}{j}$.

Let's recall that any outcome $\mathbf{x} \in \mathbf{X}$ can be viewed as a *degenerate probability distribution* that assigns all the probability to the outcome $\mathbf{x}$. Therefore, the set of all outcomes $\mathbf{X}$ is a subset of all probability distributions $\mathcal{P}$. We introduce two special probability distributions P and Q that are formed by a convex combination of certain outcomes in $\mathbf{X}$ (those outcomes are treated as degenerate probability distributions). For any fixed outcome $\mathbf{x}$, let P and Q be probability distributions defined by:

$$P = \alpha\mathbf{x} + \sum_{\substack{j \geq 2, \\ j \text{ even}}} \sum_{S \in R_j} \alpha\mathbf{x}[\cap_{s \in S} I_s],$$

$$Q = \sum_{\substack{j \geq 1, \\ j \text{ odd}}} \sum_{S \in R_j} \alpha\mathbf{x}[\cap_{s \in S} I_s],$$

$$\text{where } \alpha = \frac{1}{2^{M-1}}.$$

Both P and Q are formed from 2^{M-1} (not necessarily distinct) elements of $\mathbf{X} \subset \mathcal{P}$. In P, the sum is over all elements in R_e, and in Q, the sum is over all elements in R_o. Since

$$\sum_{S \in R_e} \alpha = \sum_{S \in R_o} \alpha = 2^{M-1} \frac{1}{2^{M-1}} = 1,$$

both P and Q are valid probability distributions in $\mathcal{P}$ (they are formed by a convex combination of other, degenerate, distributions in $\mathcal{P}$).

To define the marginal distributions P_j and Q_j, we take the marginals of the convex combination components of P and Q and use the fact that $\mathbf{x}[I]_j = \mathbf{x}_j[I]$ (the notation $\mathbf{x}_j[I]$ was defined in Section 3.1.3 above). That is, setting attributes in I^C to reference values, and then selecting out the attributes in F_j ($\mathbf{x}[I]_j$) is the same as first selecting the attributes in F_j and then setting the values of attributes in I^C to reference values ($\mathbf{x}_j[I]$). Therefore, the marginal

distributions P_j and Q_j are:

$$P_j = \alpha \mathbf{x}_j + \sum_{\substack{k \geq 2, \\ k \text{ even}}} \sum_{S \in R_k} \alpha \mathbf{x}_j [\cap_{s \in S} I_s],$$

$$Q_j = \sum_{\substack{k \geq 1, \\ k \text{ odd}}} \sum_{S \in R_k} \alpha \mathbf{x}_j [\cap_{s \in S} I_s].$$

Proof of GAI structure lemma

The GAI structure lemma 3.1 states that if the GAI condition holds, then for all $\mathbf{x} \in \mathbf{X}$:

$$u(\mathbf{x}) = \sum_{j=1}^{M} (-1)^{j+1} \sum_{\substack{S \subseteq \{1,\ldots,M\}, \\ |S|=j}} u(\mathbf{x}[\cap_{s \in S} I_s]).$$

Given two probability distributions P and Q, as defined above, it can be shown that their factor marginals are equal: $P_j = Q_j$ for all $j = 1..M$ (we provide the full proof in Appendix B). Let's assume that the GAI condition holds. Therefore, since all the marginals are equal, $P \sim Q$. By the expected utility theorem (Eq. 2.8), $P \sim Q \implies u(P) = u(Q)$, and Eq. 3.3 follows:

$$\begin{aligned}
&P_j = Q_j \text{ for all } j = 1..M && \Longrightarrow \\
&P \sim Q && \Longrightarrow \\
&u(P) = u(Q) && \Longrightarrow \\
&u\left(\alpha \mathbf{x} + \sum_{\substack{j \geq 2, \\ j \text{ even}}} \sum_{S \in R_j} \alpha \mathbf{x}[\cap_{s \in S} I_s]\right) = u\left(\sum_{\substack{j \geq 1, \\ j \text{ odd}}} \sum_{S \in R_j} \alpha \mathbf{x}[\cap_{s \in S} I_s]\right) && \Longrightarrow \\
&u(\mathbf{x}) + \sum_{\substack{j \geq 2, \\ j \text{ even}}} \sum_{S \in R_j} u(\mathbf{x}[\cap_{s \in S} I_s]) = \sum_{\substack{j \geq 1, \\ j \text{ odd}}} \sum_{S \in R_j} u(\mathbf{x}[\cap_{s \in S} I_s]) && \Longrightarrow \\
&u(\mathbf{x}) = \sum_{\substack{j \geq 1, \\ j \text{ odd}}} \sum_{S \in R_j} u(\mathbf{x}[\cap_{s \in S} I_s]) - \sum_{\substack{j \geq 2, \\ j \text{ even}}} \sum_{S \in R_j} u(\mathbf{x}[\cap_{s \in S} I_s]) && \Longrightarrow \\
&u(\mathbf{x}) = \sum_{j=1}^{M} (-1)^{j+1} \sum_{S \in R_j} u(\mathbf{x}[\cap_{s \in S} I_s]).
\end{aligned}$$

This proves the GAI structure lemma 3.1. □

3.1.5 Proof of GAI representation theorem

The GAI structure lemma 3.1 is a key element in the proof of the GAI representation theorem. The GAI representation theorem 3.1 states that the GAI condition holds if and only if there exist functions $u_1, \ldots, u_M$ on $\mathcal{P}_1, \ldots, \mathcal{P}_M$ such that $u(P) = \sum_{j=1}^{M} u_j(P_j)$.

"If" direction

We first show that having a utility function of the form $u(P) = \sum_{j=1}^{M} u_j(P_j)$ implies the GAI condition, i.e., if two probability distributions P and Q have the same marginals on factors, $P \sim Q$. Let's assume P and Q have equal marginals on factors: $P_j = Q_j$ for all $j = 1..M$. This means that their utilities have to be equal, too: $u(P) = \sum_{j=1}^{M} u_j(P_j) = \sum_{j=1}^{M} u_j(Q_j) = u(Q)$. By the expected utility theorem (Eq. 2.8), this implies that $P \sim Q$. $\square$

"Only if" direction

The other direction of the proof is more complicated, and relies on the GAI structure lemma 3.1. Here, we have to prove that the GAI condition implies the existence of subutility functions $u_1, \ldots, u_M$ such that $u(\mathbf{x}) = \sum_{j=1}^{M} u_j(\mathbf{x}_j)$.

We start by defining subutility functions u_j on $\mathbf{X}_j$:

$$u_j(\mathbf{x}_j) = u(\mathbf{x}[I_j]) + \sum_{1 \leq q < j} (-1)^q \sum_{\substack{S \subseteq \{1,\ldots,j-1\}, \\ |S|=q}} u(\mathbf{x}[\bigcap_{s \in S} I_s \cap I_j]). \tag{3.4}$$

The subutility function u_j is well-defined since for $\mathbf{x}, \mathbf{y} \in \mathbf{X}$, $u_j(\mathbf{x}_j) = u_j(\mathbf{y}_j)$ if $\mathbf{x}_j = \mathbf{y}_j$. This is because all outcomes on the right-hand side are basic outcomes of factor F_j (that is, in all outcomes on the right-hand side, the attributes that are not in factor F_j are set to their reference

values). Summing over all factors:

$$\begin{aligned}
\sum_{j=1}^{M} u_j(\mathbf{x}_j) &= \sum_{j=1}^{M} u(\mathbf{x}[I_j]) + \sum_{j=1}^{M} \sum_{1\le q<j} (-1)^q \sum_{\substack{S\subseteq\{1,\ldots,j-1\},\\ |S|=q}} u(\mathbf{x}[\bigcap_{s\in S} I_s \cap I_j]) \\
&= \sum_{j=1}^{M} u(\mathbf{x}[I_j]) + \sum_{q=1}^{M-1} (-1)^q \sum_{j=q+1}^{M} \sum_{\substack{S\subseteq\{1,\ldots,j-1\},\\ |S|=q}} u(\mathbf{x}[\bigcap_{s\in S} I_s \cap I_j]) \\
&= \sum_{j=1}^{M} u(\mathbf{x}[I_j]) + \sum_{q=1}^{M-1} (-1)^q \sum_{\substack{S\subseteq\{1,\ldots,M\},\\ |S|=q+1}} u(\mathbf{x}[\bigcap_{s\in S} I_s]) \\
&= \sum_{\substack{S\subseteq\{1,\ldots,M\},\\ |S|=1}} u(\mathbf{x}[\bigcap_{s\in S} I_s]) + \sum_{j=2}^{M} (-1)^{j+1} \sum_{\substack{S\subseteq\{1,\ldots,M\},\\ |S|=j}} u(\mathbf{x}[\bigcap_{s\in S} I_s]) \\
&= \sum_{j=1}^{M} (-1)^{j+1} \sum_{\substack{S\subseteq\{1,\ldots,M\},\\ |S|=j}} u(\mathbf{x}[\bigcap_{s\in S} I_s]).
\end{aligned}$$

We now know that $\sum_{j=1}^{M} u_j(\mathbf{x}_j) = \sum_{j=1}^{M}(-1)^{j+1} \sum_{S\subseteq\{1,\ldots,M\},|S|=j} u(\mathbf{x}[\bigcap_{s\in S} I_s])$. We assume that the GAI condition holds. Therefore, according to the GAI structure lemma 3.1, $u(\mathbf{x}) = \sum_{j=1}^{M}(-1)^{j+1} \sum_{S\subseteq\{1,\ldots,M\},|S|=j} u(\mathbf{x}[\bigcap_{s\in S} I_s])$, which means that

$$\sum_{j=1}^{M} u_j(\mathbf{x}_j) = \sum_{j=1}^{M} (-1)^{j+1} \sum_{\substack{S\subseteq\{1,\ldots,M\},\\ |S|=j}} u(\mathbf{x}[\bigcap_{s\in S} I_s]) = u(\mathbf{x}).$$

Thus, $u(\mathbf{x}) = \sum_{j=1}^{M} u_j(\mathbf{x}_j)$, which concludes the proof of the GAI representation theorem.□

3.2 Local structure semantics in GAI models

In this section, we discuss problems that arise in eliciting and reasoning with GAI subutility functions, and propose specific solutions to these issues. In particular, we show how to interpret the GAI subutility semantics, explore the graphical nature of GAI models, and present a way to write the model in a manner that preserves the local value function semantics of additive models. Using our representation, GAI models can be elicited by largely using local queries about preferences over small subsets of attributes, and relying on global queries only for calibration

across utility factors. This allows one to exploit the generality, compactness and much wider applicability of GAI models without losing the advantage of elicitation based on local queries (offered by more restrictive additive models).

3.2.1 GAI subutility functions

3.2.1.1 Additive utilities

Before we take a deeper look at the subutility functions in GAI models, let's consider the simpler additive model. Similar to the GAI independence condition, additive independence leads to the following additive utility representation (Eq. 2.14):

$$u(\mathbf{x}) = \sum_{i=1}^{N} u_i(x_i).$$

The domains of subutility functions are single attributes, rather than sets of attributes, as in the case of general GAI utilities.

For a given additive utility function u, the subutilities u_i have a very special property: they are *unique up to the simultaneous transformations* $u'_i(x_i) = u_i(x_i) + c_i$ *for all* $x_i \in X_i$, $i = 1, 2, \ldots, N$, $\sum c_i = 0$ (Fishburn, 1965). That is, u could be written as a sum of functions u_i, or a sum of functions u'_i, but the only allowable difference between u_i and u'_i is the constant c_i.

This is a very strong condition on the possible transformations of the subutility functions. Due to this condition, the subutility functions u_i for additive utilities preserve local preference relations, which is a very intuitive and desirable property. That is,

$$\begin{aligned} x_i \succeq y_i, \text{ ceteris paribus } &\iff u_i(x_i) \geq u_i(y_i) \\ &\iff u_i(x_i) + c_i \geq u_i(y_i) + c_i \\ &\iff u'_i(x_i) \geq u'_i(y_i) \text{ for some } u'_i. \end{aligned} \tag{3.5}$$

Thus, additive subutility functions are essentially *unique* (apart from an additive constant c_i) and they *represent local preference relations*.

3.2.1.2 Non-uniqueness of GAI subutilities

Unlike additive models, GAI subutilities are *not unique*; in fact, for any GAI utility function with a fixed set of overlapping factors, there are an infinite number of non-trivial ways to define the subutility functions u_j (Eq. 3.2) such that they still comprise the same utility function u. Furthermore, unlike in additive models, the values of subutility functions u_j do not directly represent the local preference relation among the attributes in factor F_j. Since the utility can "flow" from one subutility factor to the next through the shared attributes, the subutility values do not have an independent semantic meaning.

The following examples illustrate the problem.

Example 3.4 In the additive case, $u_i(x_i^1) > u_i(x_i^2)$ implies that outcomes with i^{th} attribute set to x_i^1 are preferred to outcomes with x_i^2, as long as the rest of attributes are kept constant. However, this implication does not extend to GAI models. Let's take our three-attribute example $u(x, y, z) = u_1(x, y) + u_2(y, z)$ from Table 3.2. If we knew that $u_1(x^1, y^1) = 100$ and $u_1(x^2, y^2) = 50$, would that imply $(x^1, y^1) \succeq (x^2, y^2)$, ceteris paribus? Based on the u_1 parameters, it would seem natural that the highest valued outcome in u_1, (x^1, y^1), would be preferred to the lowest valued outcome (x^2, y^2), ceteris paribus. As we shall see, this is not the case. It turns out that because of factor interdependence through shared attributes, we can rewrite the utility function u as follows

$$\begin{aligned} u(x, y, z) &= u_1(x, y) + u_2(y, z) \\ &= [u_1(x, y) + f(y)] + [u_2(y, z) - f(y)] \\ &= u_1'(x, y) + u_2'(y, z), \end{aligned}$$

where $f : Y \mapsto \mathbb{R}$ is an arbitrary real-valued function.

By setting $f(y^1) = -50$, and $f(y^2) = 50$, we obtain $u_1'(x^1, y^1) = 50$ and $u_1'(x^2, y^2) = 100$, the exact opposite values from those in $u_1(\cdot)$. The subutility value changes in the first factor are suitably compensated by changes in the second factor, leaving the overall utility function exactly the same. Since the utility can "flow" from one subutility factor to the next through

$\mathbf{x}_1$	$u_1(\mathbf{x}_1)$	$u'_1(\mathbf{x}_1)$
x^1y^1	**100**	**50**
x^2y^1	70	20
x^1y^2	90	140
x^2y^2	**50**	**100**

(a) Factor F_1

$\mathbf{x}_2$	$u_2(\mathbf{x}_2)$	$u'_2(\mathbf{x}_2)$
y^1z^1	100	150
y^2z^1	50	0
y^1z^2	60	110
y^2z^2	30	-20

(b) Factor F_2

$\mathbf{x}$	$u_1(\mathbf{x}_1) + u_2(\mathbf{x}_2)$	$u'_1(\mathbf{x}_1) + u'_2(\mathbf{x}_2)$	$u(\mathbf{x})$
$x^1y^1z^1$	100+100	50+150	200
$x^2y^1z^1$	70+100	20+150	170
$x^1y^2z^1$	90+50	140+0	140
$x^2y^2z^1$	50+50	100+0	100
$x^1y^1z^2$	100+60	50 + 110	160
$x^2y^1z^2$	70+60	20 + 110	130
$x^1y^2z^2$	90+30	140-20	120
$x^2y^2z^2$	50+30	100-20	80

(c) GAI utility function

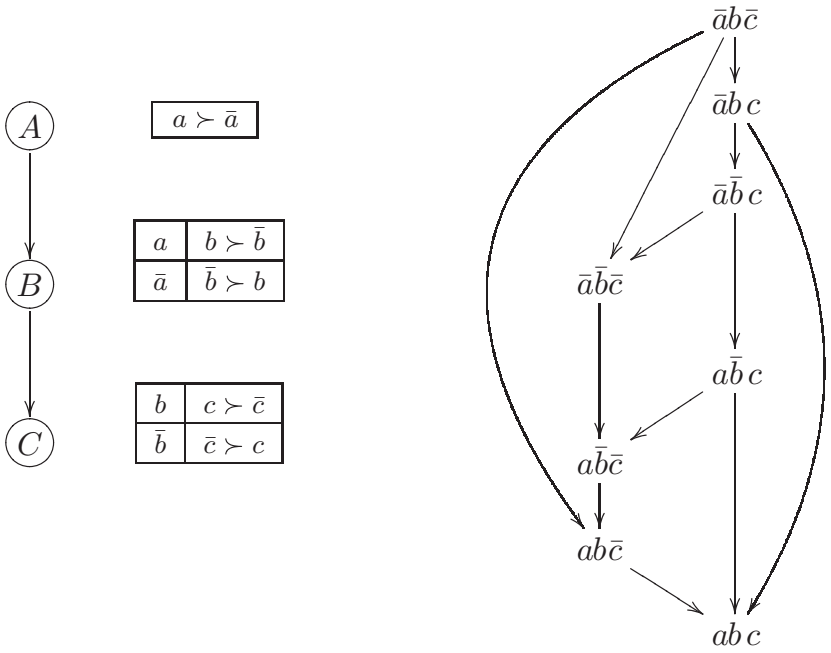
Table 3.2: The parameters of the GAI utility function $u(x, y, z) = u_1(x, y) + u_2(y, z) = u'_1(x, y) + u'_2(y, z)$ from Example 3.4. In the first decomposition, $u_1(x^1, y^1) > u_1(x^2, y^2)$; however, in the second decomposition $u'_1(x^1, y^1) < u'_1(x^2, y^2)$. The semantics of subutility functions is therefore not obvious.

the shared attributes, the subutility values do not have an independent semantic meaning. In GAI settings, local factor outcomes cannot be compared in a "ceteris paribus" manner (as in the additive utility case), *because the local order depends on the particular instantiation of some of the attributes outside the factor* (the smallest set of such attributes will be later called the *conditioning set*). We can also see that the same utility function can be decomposed in an infinite number of non-trivial ways (because f is an arbitrary real-valued function). Table 3.2 shows all the parameters of the three-attribute utility function used in this example.

Example 3.5 The second example shows that possibility of non-trivial transformations of GAI subutility functions is problematic for the UCP network semantics. A UCP network (Boutilier et al., 2001) is a graphical representation of a GAI utility such that its network structure reflects conditional dependencies of an underlying CP network (Boutilier et al., 1999, 2004b), and the represented utility function satisfies all of the conditional preference statements of the CP network. Figure 3.1 shows a simple CP network (Boutilier et al., 2004b), with its two UCP extensions (Figures 3.1b, 3.1c). Both UCP extensions represent the same utility function that induces the following preference order satisfying the underlying CP network:

$$abc \succeq ab\bar{c} \succeq a\bar{b}\bar{c} \succeq a\bar{b}c \succeq \bar{a}\bar{b}\bar{c} \succeq \bar{a}\bar{b}c \succeq \bar{a}bc \succeq \bar{a}b\bar{c}$$

However, the second extension, obtained from the first by setting $f(b) = 5, f(\bar{b}) = 30$, results in UCP factor subutilities that do not allow us to *directly* recover corresponding conditional preference tables (CPTs) of the underlying CP network. For example, we know that, given a, $b \succ \bar{b}$, ceteris paribus. As expected, in the first UCP network, $u_1(ab) > u_2(a\bar{b})$, but in the second, $u'_1(ab) < u'_2(a\bar{b})$. Both networks are valid UCP representations, since they define the same utility function, and satisfy all of the conditional preference statements of the underlying CP network. However, because the second network does not allow us to directly read off the underlying CP preferences, the elicitation of local factor subutilities is more involved than the method proposed in the original UCP paper (Boutilier et al., 2001).



(a) CP-net, with conditional preference tables and its partial order graph

	$u_0(A)$		$u_1(A,B)$		$u_2(B,C)$
a	5	ab	35	bc	10
$\bar{a}$	0	$a\bar{b}$	15	$b\bar{c}$	8
		$\bar{a}\bar{b}$	12	$\bar{b}\bar{c}$	2
		$\bar{a}b$	2	$\bar{b}c$	1

(b) UCP extension 1

	$u'_0(A)$		$u'_1(A,B)$		$u'_2(B,C)$
a	5	ab	40	bc	5
$\bar{a}$	0	$a\bar{b}$	45	$b\bar{c}$	3
		$\bar{a}\bar{b}$	42	$\bar{b}\bar{c}$	-28
		$\bar{a}b$	7	$\bar{b}c$	-29

(c) UCP extension 2

Figure 3.1: A CP network (Boutilier et al., 2004b) and two equivalent UCP extensions (see Example 3.5 for full details).

3.2.1.3 Admissible transformations

To get a better insight into the problem, we need to answer the following question: given a utility function $u(\mathbf{x}) = \sum u_j(\mathbf{x}_j)$, what are the possible transformations on subutility functions u_j such that u remains unchanged? As we noted before, for additive utilities, such *admissible* transformations are (Fishburn, 1965):

$$u'_i(x_i) = u_i(x_i) + c_i, \text{ for all } i = 1, \ldots, N,$$
$$\sum_{i=1}^{N} c_i = 0.$$

For GAI utilities, admissible transformations are more general. For any two factors F_j and F_k with shared attributes, we can define an arbitrary function $f_{jk} : \mathbf{X}_{I_j \cap I_k} \mapsto \mathbb{R}$ on all the configurations of the shared attributes, and transform u_j by adding f_{jk}, and u_k by subtracting f_{jk}. Such transformations can drastically change *local* subutility functions, without changing the overall utility function. More formally, admissible transformations for GAI models are (Fishburn, 1967b):

$$u'_j(\mathbf{x}_j) = u_j(\mathbf{x}_j) + \sum_{\substack{j \neq k \\ I_j \cap I_k \neq \emptyset}} f_{jk}(\mathbf{x}_{I_j \cap I_k}) + c_j, \text{ for all } j = 1, \ldots, M, \tag{3.6}$$
$$\sum_{j=1}^{M} \left[\sum_{\substack{j \neq k \\ I_j \cap I_k \neq \emptyset}} f_{jk}(\mathbf{x}_{I_j \cap I_k}) + c_j \right] = 0.$$

Since the admissible transformations for GAI utilities are more general than positive affine transformations, subutility functions u_j and u'_j are not necessarily strategically equivalent (i.e., they do not necessarily represent the same preference order over local lotteries). The rest of the chapter describes a semantically sound way of dealing with this problem.

3.2.1.4 Canonical subutilities

The key difference between additive and GAI models with regard to elicitation (rather than representation) lies in the *semantics* of subutility functions u_j. In additive models, subutilities

u_j have a very clear decision-theoretic meaning. In contrast, GAI subutility functions *are not unique* and, in the absence of further qualifications, do not have a well-defined semantic interpretation. To solve this issue, we designate a particular GAI subutility representation as the *canonical* representation, and devise elicitation queries and techniques that preserve the semantics of GAI models.

As we saw in Section 3.1.2, GAI subutility functions are constructed by a judicious grouping of addends on the right side of the Eq. 3.3 (GAI structure lemma), resulting in functions $u_1, \ldots, u_M$ such that $u(\mathbf{x}) = \sum_{j=1}^{M} u_j(\mathbf{x}_j)$:

$$u(\mathbf{x}) = \sum_{j=1}^{M} (-1)^{j+1} \sum_{\substack{S \subseteq \{1,\ldots,M\}, \\ |S|=j}} u(\mathbf{x}[\cap_{s \in S} I_s]).$$

Given a GAI model, however, there are multiple ways of defining subutility functions. For example, with factors $F_1 = \{X_1, X_2\}$ and $F_2 = \{X_2, X_3\}$, we have:

$$u(x_1, x_2, x_3) = u(x_1, x_2, x_3^0) + u(x_1^0, x_2, x_3) - u(x_1^0, x_2, x_3^0).$$

One way to define subutility functions would be to group $u(x_1, x_2, x_3^0)$ and $u(x_1^0, x_2, x_3^0)$, so that $u_1(x_1, x_2) = u(x_1, x_2, x_3^0) - u(x_1^0, x_2, x_3^0)$, and $u_2(x_2, x_3) = u(x_1^0, x_2, x_3)$. Another subutility decomposition would result if instead $u(x_1^0, x_2, x_3)$ and $u(x_1^0, x_2, x_3^0)$ were put together: $u_1'(x_1, x_2) = u(x_1, x_2, x_3^0)$, and $u_2'(x_2, x_3) = u(x_1^0, x_2, x_3) - u(x_1^0, x_2, x_3^0)$.

To define a *canonical* representation of GAI subutility functions, we adopt the grouping used by Fishburn (1967b) in proving the GAI representation theorem. Consider the set $D = \{(-1)^{|S|+1} u(\mathbf{x}[\cap_{s \in S} I_s]), \ S \subseteq \{1, \ldots, M\}\}$ of all the addends that appear on the right side of the equation above (same as Eq. 3.3). Each addend corresponds to some subset $S \subseteq \{1, \ldots, M\}$. We partition the set D into M groups as follows: an addend corresponding to the subset S belongs to group j if j is the greatest element in S. For example, with three factors F_1, F_2, F_3, addends $u(\mathbf{x}[I_2])$ and $-u(\mathbf{x}[I_1 \cap I_2])$ belong to group 2, whereas $-u(\mathbf{x}[I_1 \cap I_3])$ and $u(\mathbf{x}([I_1 \cap I_2 \cap I_3])$ belong to group 3.

By rewriting the order of addends, we obtain a canonical GAI subutility decomposition:

$$\begin{aligned} u(\mathbf{x}) &= \sum_{j=1}^{M} (-1)^{j+1} \sum_{\substack{S\subseteq\{1,\ldots,M\},\\ |S|=j}} u(\mathbf{x}[\cap_{s\in S} I_s]) \\ &= \sum_{j=1}^{M} \left[u(\mathbf{x}[I_j]) + \sum_{1\le q<j} (-1)^q \sum_{\substack{S\subseteq\{1,\ldots,j-1\},\\ |S|=q}} u(\mathbf{x}[\bigcap_{s\in S} I_s \cap I_j]) \right] \\ &= \sum_{j=1}^{M} u_j(\mathbf{x}_j). \end{aligned}$$

Definition 3.3 (Canonical GAI subutility decomposition)

$$\begin{aligned} u_1(\mathbf{x}_1) &= u(\mathbf{x}[I_1]), \\ u_j(\mathbf{x}_j) &= u(\mathbf{x}[I_j]) + \sum_{1\le q<j} (-1)^q \sum_{\substack{S\subseteq\{1,\ldots,j-1\},\\ |S|=q}} u(\mathbf{x}[\bigcap_{s\in S} I_s \cap I_j]). \end{aligned} \tag{3.7}$$

Example 3.6 With three factors F_1, F_2, F_3, addends $u(\mathbf{x}[I_2])$ and $u(\mathbf{x}[I_1 \cap I_2])$ belong to Group 2, whereas $u(\mathbf{x}[I_1 \cap I_3])$ and $u(\mathbf{x}([I_1 \cap I_2 \cap I_3])$ belong to Group 3:

$$\begin{aligned} u(\mathbf{x}) &= u(\mathbf{x}[I_1]) + u(\mathbf{x}[I_2]) + u(\mathbf{x}[I_3]) \\ &\quad - u(\mathbf{x}[I_1 \cap I_2]) - u(\mathbf{x}[I_1 \cap I_3]) - u(\mathbf{x}[I_2 \cap I_3]) \\ &\quad + u(\mathbf{x}[I_1 \cap I_2 \cap I_3]) \\ &= u(\mathbf{x}[I_1]) \\ &\quad + u(\mathbf{x}[I_2]) - u(\mathbf{x}[I_1 \cap I_2]) \\ &\quad + u(\mathbf{x}[I_3]) - [u(\mathbf{x}[I_1 \cap I_3]) + u(\mathbf{x}[I_2 \cap I_3])] + u(\mathbf{x}[I_1 \cap I_2 \cap I_3]). \end{aligned}$$

Thus,

$$\begin{aligned} u_1(\mathbf{x}_1) &= u(\mathbf{x}[I_1]), \\ u_2(\mathbf{x}_2) &= u(\mathbf{x}[I_2]) - u(\mathbf{x}[I_1 \cap I_2]), \\ u_3(\mathbf{x}_3) &= u(\mathbf{x}[I_3]) - u(\mathbf{x}[I_1 \cap I_3]) - u(\mathbf{x}[I_2 \cap I_3]) + u(\mathbf{x}[I_1 \cap I_2 \cap I_3]). \end{aligned}$$

We should keep in mind that u denotes utility of full outcomes, whereas u_j is defined only over attributes in factor F_j.

3.2.2 GAI structure

Having proposed the canonical decomposition of GAI models, in this section we investigate the properties of subutility functions u_j and algorithms for their computation.

Relevant and irrelevant subsets

To better understand how to compute a subutility function $u_j(\mathbf{x}_j)$, we rewrite it as follows:

$$\begin{aligned} u_j(\mathbf{x}_j) &= u(\mathbf{x}[I_j]) + \sum_{1\le q<j} (-1)^q \sum_{\substack{S_q\subseteq\{1,\ldots,j-1\},\\ |S_q|=q}} u(\mathbf{x}[\bigcap_{s\in S_q} I_s \cap I_j]) \\ &= u(\mathbf{x}[I_j]) + \sum_{\substack{S\subseteq\{1,\ldots,j-1\},\\ S\neq\emptyset}} (-1)^{|S|}\, u(\mathbf{x}[\bigcap_{s\in S} I_s \cap I_j]). \end{aligned} \tag{3.8}$$

All the outcomes on the right-hand side are basic outcomes for factor F_j. Eq. 3.8 relates factor subutility values to utilities of its basic outcomes.

Observation 3.4 *The number of addends in Eq. 3.8 above is 2^{j-1}. The number of distinct addends (ignoring sign coefficients $(-1)^{|S|}$) is at most $N_j = |\mathbf{X}_j|$, the number of all local configurations in factor F_j.*

Proof The summation above is over all 2^{j-1} members of the powerset of $\{1,\ldots,j-1\}$, except $\emptyset$. Thus, the total number of addends is $1 + (2^{j-1} - 1) = 2^{j-1}$. As observed before, all the outcomes on the right-hand side of Eq. 3.8 are basic outcomes for factor F_j, and each factor has N_j basic outcomes.□

Our goal is to find the basic outcomes involved in Eq. 3.8, which involves summing over all the subsets of $\{1,\ldots,j-1\}$. In practice, we expect GAI models to exhibit considerable structure, and intersections between factors to involve only a few variables. In such case, for a lot of subsets $S \subseteq \{1,\ldots,j-1\}$, the intersections $\bigcap_{s\in S} I_s$ (and, therefore, $(\bigcap_{s\in S} I_s)\cap I_j$), will

be empty. We denote such subsets S as *irrelevant* (for factor F_j), and the ones whose member intersections with I_j are non-empty as *relevant*. By separating the summations over relevant and irrelevant sets, we have:

$$u_j(\mathbf{x}_j) = u(\mathbf{x}[I_j]) + \sum_{\substack{S\subseteq\{1,\ldots,j-1\},\\ S\neq\emptyset}} (-1)^{|S|}\; u(\mathbf{x}[\bigcap_{s\in S} I_s \cap I_j])$$

$$= u(\mathbf{x}[I_j]) + \sum_{\substack{S\subseteq\{1,\ldots,j-1\},\\ S \text{ is relevant}}} (-1)^{|S|}\; u(\mathbf{x}[\bigcap_{s\in S} I_s \cap I_j]) + \sum_{\substack{S\subseteq\{1,\ldots,j-1\},\\ S \text{ is irrelevant}}} (-1)^{|S|}\; u(\mathbf{x}^0).$$

Here, we use the fact that $\mathbf{x}[\emptyset] = \mathbf{x}^0$.

Let σ_R denote the sum of coefficients $(-1)^{|S|}$ over all relevant sets S, and $\sigma_{\bar{R}}$ denote the sum of coefficients $(-1)^{|S|}$ over all irrelevant sets S:

$$\sigma_R = \sum_{\substack{S\subseteq\{1,\ldots,j-1\},\\ S \text{ is relevant}}} (-1)^{|S|}$$

$$\sigma_{\bar{R}} = \sum_{\substack{S\subseteq\{1,\ldots,j-1\},\\ S \text{ is irrelevant}}} (-1)^{|S|}. \tag{3.9}$$

Then, since the third part of the subutility equation above can be written as

$$\sum_{\substack{S\subseteq\{1,\ldots,j-1\},\\ S \text{ is irrelevant}}} (-1)^{|S|}\; u(\mathbf{x}^0) = u(\mathbf{x}^0) \sum_{\substack{S\subseteq\{1,\ldots,j-1\},\\ S \text{ is irrelevant}}} (-1)^{|S|} = \sigma_{\bar{R}}\, u(\mathbf{x}^0),$$

the subutility equation becomes:

$$u_j(\mathbf{x}_j) = u(\mathbf{x}[I_j]) + \sum_{\substack{S\subseteq\{1,\ldots,j-1\},\\ S \text{ is relevant}}} (-1)^{|S|}\; u(\mathbf{x}[\bigcap_{s\in S} I_s \cap I_j]) + \sigma_{\bar{R}}\, u(\mathbf{x}^0). \tag{3.10}$$

Eq. 3.10 is useful, because it shows that to compute a GAI subutility function, we only need to consider relevant subsets S (the number of which, if GAI factors do not overlap too much, will be small), and know the coefficient $\sigma_{\bar{R}}$. Fortunately, computing $\sigma_{\bar{R}}$ is easy, once the collection of relevant sets (and, therefore, σ_R) is known.

Observation 3.5

$$\sigma_{\bar{R}} = -1 - \sigma_R = -1 - \sum_{\substack{S\subseteq\{1,\ldots,j-1\},\\ S \textit{ is relevant}}} (-1)^{|S|}, \textit{ for } j \geq 2.$$

When $j = 1$, $\sigma_{\bar{R}} = 0$.

Proof We can prove this by using binomial expansion (similar to the previous result for Observation 3.3). For $j \geq 2$:

$$1 + \sigma_R + \sigma_{\bar{R}} = 1 + \sum_{\substack{S \subseteq \{1,\ldots,j-1\},\\ S \neq \emptyset}} (-1)^{|S|} = \sum_{k=0}^{j-1} \binom{j-1}{k} (-1)^k = (1-1)^{j-1} = 0.$$

When $j = 1$, the above equation is equal to 1, instead of 0. In this special case, $\sigma_R = 1$, and $\sigma_{\bar{R}} = 0$. $\square$

Let L_j denote the set of all relevant sets for factor F_j:

$$\begin{aligned} L_j &= \{S | S \subseteq \{1, \ldots, j-1\},\ S \text{ is relevant}\} \\ &= \{S | S \subseteq \{1, \ldots, j-1\},\ \bigcap_{s \in S} I_s \cap I_j \neq \emptyset\} \\ &= \{S | S \subseteq \{k | k < j, I_k \cap I_j \neq \emptyset\},\ \bigcap_{s \in S} I_s \cap I_j \neq \emptyset\}. \end{aligned} \tag{3.11}$$

In the equation above, the second line follows from the definition of relevant sets. The third line provides an alternative definition of relevant sets by taking into account the observation that any element s of a relevant set S has to satisfy the condition $I_s \cap I_j \neq \emptyset$ (since, by definition, relevant sets satisfy the stronger condition $\bigcap_{s \in S} I_s \cap I_j \neq \emptyset$).

Eq. 3.10 can now be rewritten as:

$$\begin{aligned} u_j(\mathbf{x}_j) &= u(\mathbf{x}[I_j]) + \sum_{\substack{S \subseteq \{1,\ldots,j-1\},\\ S \text{ is relevant}}} (-1)^{|S|}\ u(\mathbf{x}[\bigcap_{s \in S} I_s \cap I_j]) + \sigma_{\bar{R}}\, u(\mathbf{x}^0) \\ &= u(\mathbf{x}[I_j]) + \sum_{\substack{S \subseteq \{k | k < j, I_k \cap I_j \neq \emptyset\},\\ S \text{ is relevant}}} (-1)^{|S|}\ u(\mathbf{x}[\bigcap_{s \in S} I_s \cap I_j]) + \sigma_{\bar{R}}\, u(\mathbf{x}^0) \\ &= u(\mathbf{x}[I_j]) + \sum_{S \in L_j} (-1)^{|S|}\ u(\mathbf{x}[\bigcap_{s \in S} I_s \cap I_j]) + \sigma_{\bar{R}}\, u(\mathbf{x}^0). \end{aligned} \tag{3.12}$$

In the summation above, instead of considering all the possible 2^{j-1} subsets of $\{1, \ldots, j-1\}$, we only have to consider subsets of indices of factors preceding F_j that share attributes with factor F_j. In GAI models with limited dependencies among factors, this significantly reduces the complexity of finding the relevant sets. Below, we describe a graphical search procedure for finding the relevant sets that is based on Eq. 3.12.

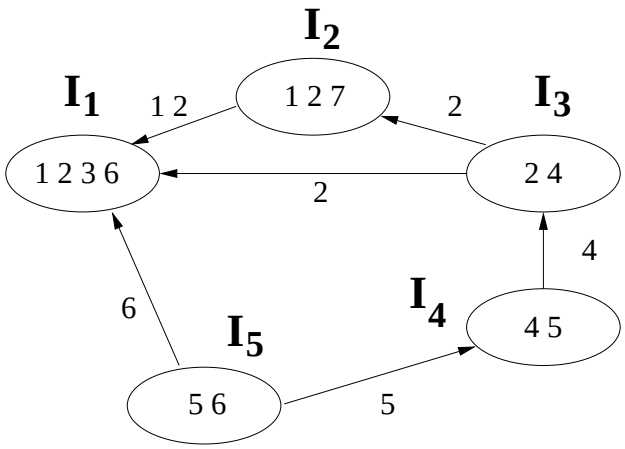


Figure 3.2: An example of a GAI graph. Nodes correspond to factor index sets, and higher index nodes link to lower index nodes if the intersection between the factor index sets is not empty. Edges are labeled by the intersections of the adjacent node index sets.

Graphical search procedure for finding relevant sets

We assume a set of factors $\{F_1, ..., F_M\}$ with an arbitrary ordering induced by the M subscripts. Each factor F_j has an associated index set I_j denoting the indices of the attributes in F_j: $F_j = \{X_i \mid i \in I_j\}$. The index sets I_j cover all attribute indices: $\cup I_j = \{1, \ldots, N\}$.

Given the factors, we construct a *GAI graph* whose nodes are numbered from 1 to M. The node i corresponds to the index set I_i; two nodes j and k are linked by a directed edge from j to k if $I_k \cap I_j \neq \emptyset$ (that is, the index sets I_k and I_j share some variables) and $k < j$. Since directed edges always link higher-numbered nodes to lower-numbered nodes, there can be no cycles: a GAI graph is a directed acyclic graph.[1] Fig. 3.2 shows an example of a GAI graph, with edges labeled with intersections of adjacent node index sets.

A GAI graph can be used to compute the relevant sets for GAI factors. For a given factor F_j, any node $k < j$ that intersects the starting node j can be reached by following the directed edges from j to k. Furthermore, all the nodes along any path from j to k also intersect the starting node j. If a node does not intersect with j, then its descendants do not intersect with j either. Each relevant set for factor F_j corresponds to the nodes along a distinct path from j to

[1] An undirected version of the GAI graph was called a *GAI network* by Gonzales and Perny (2004). However, the GAI network was used for a different purpose.

Input: GAI attribute sets $I_1, \ldots, I_M$
Output: For each subutility factor F_j, a collection L_j of relevant sets

foreach *factor* F_j **do**
  $L_j = \emptyset$
  Start at node j and perform depth-first search along the directed arcs
  foreach *node* k **do**
    $S = \{$nodes on path from j to $k\}$
    if $\bigcap_{s \in S} I_s \cap I_j \neq \emptyset$*, i.e., the intersection of all the nodes on the current path from* j *to* k *is non-empty* **then**
      add S to L_j
      visit the children nodes of node k
    end
  end
end

Figure 3.3: Algorithm for computing relevant sets for GAI factors

some node k whose intersection with j is non-empty.

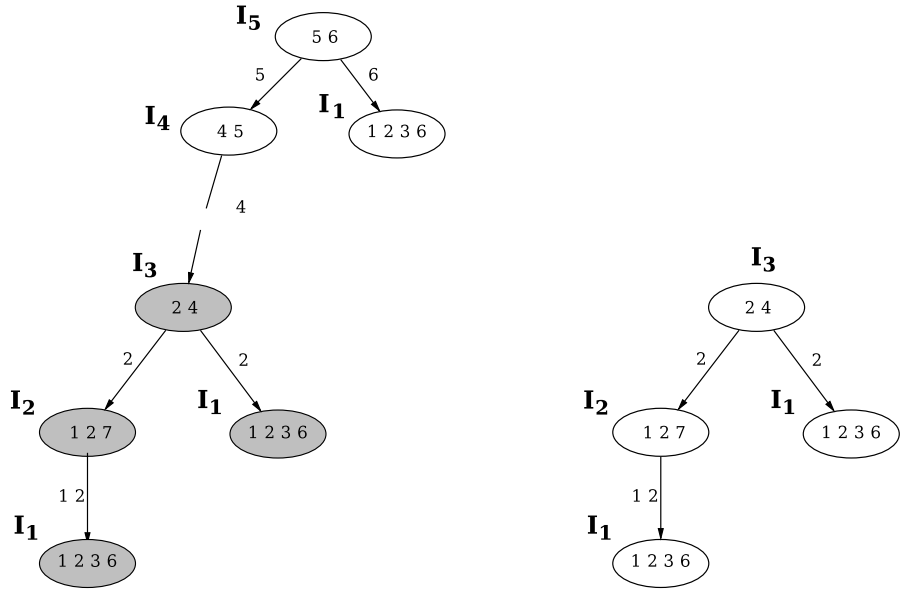
The full algorithm to compute the relevant sets for all factors is shown in Figure 3.3. For each factor F_j, we start with the node j, and search along directed edges for all lower-numbered nodes that intersect with the starting node j. If some node k intersects j, the set of all nodes along the path from j to k is a relevant set. If k does not intersect j, then we can backtrack, because the descendants of k do not intersect j either. We store all the relevant sets for factor F_j in the collection L_j.

After finding the relevant sets $L_1, \ldots, L_j$, we can define subutility functions using Eq. 3.12:

$$u_j(\mathbf{x}_j) = u(\mathbf{x}[I_j]) + \sum_{S \in L_j} (-1)^{|S|}\, u(\mathbf{x}[\bigcap_{s \in S} I_s \cap I_j]) + \sigma_{\bar{R}}\, u(\mathbf{x}^0),$$

where $\sigma_{\bar{R}} = -1 - \sum_{S \in L_j} (-1)^{|S|}$, unless $j = 1$, in which case it is 0.

Example 3.7 Consider the GAI graph in Fig. 3.2 and the search trees for factors F_5 and F_3 in Fig. 3.4(a) and Fig. 3.4(b). To compute L_5, we search along the directed edges for all non-empty intersections of the set I_5 with other sets. The only such sets are I_4 (depth 1) and I_1 (depth 1). Therefore, $L_5 = \{\{4\}, \{1\}\}$, and $u_5(\mathbf{x}_5) = u(\mathbf{x}[\{I_5\}]) - u(\mathbf{x}[\{I_4 \cap I_5\}]) - u(\mathbf{x}[\{I_1 \cap I_5\}]) + u(\mathbf{x}^0) = u(\mathbf{x}[\{5,6\}]) - u(\mathbf{x}[\{5\}]) - u(\mathbf{x}[\{6\}]) + u(\mathbf{x}^0)$. Similarly, the relevant sets for factor F_3 are $L_3 = \{\{2\}, \{2,1\}, \{1\}\}$, and $u_3(\mathbf{x}_3) = u(\mathbf{x}[\{I_3\}]) - u(\mathbf{x}[\{I_2 \cap I_3\}]) +$



(a) Search tree for factor F_5 (b) Search tree for factor F_3

Figure 3.4: The search trees of a GAI graph from Figure 3.2 for finding the relevant sets for factors F_5 (3.4a) and F_3 (3.4a). Each node in a search tree corresponds to a relevant set consisting of the node and all the preceding nodes. If a node's intersection with the root node is empty, the subtree rooted at that node does not have to be searched (in Figure 3.4a, the shaded subtree rooted at I_3 is not relevant, since the nodes in that subtree do not intersect I_5). To find the relevant sets, we visit every node in a search tree and record the preceding nodes (except the root node). For F_5 (3.4a), $L_5 = \{\{4\}, \{1\}\}$; therefore, $u_5(\mathbf{x}_5) = u(\mathbf{x}[\{I_5\}]) - u(\mathbf{x}[\{I_4 \cap I_5\}]) - u(\mathbf{x}[\{I_1 \cap I_5\}]) + u(\mathbf{x}^0) = u(\mathbf{x}[\{5,6\}]) - u(\mathbf{x}[\{5\}]) - u(\mathbf{x}[\{6\}]) + u(\mathbf{x}^0)$. Similarly, the relevant sets for factor F_3 are $L_3 = \{\{2\}, \{2,1\}, \{1\}\}$, and $u_3(\mathbf{x}_3) = u(\mathbf{x}[\{I_3\}]) - u(\mathbf{x}[\{I_2 \cap I_3\}]) + u(\mathbf{x}[\{I_2 \cap I_1 \cap I_3\}]) - u(\mathbf{x}[\{I_1 \cap I_3\}]) = u(\mathbf{x}[\{2,4\}]) - u(\mathbf{x}[\{2\}]) + u(\mathbf{x}[\{2\}]) - u(\mathbf{x}[\{2\}]) = u(\mathbf{x}[\{2,4\}]) - u(\mathbf{x}[\{2\}])$. Example 3.7 provides further details.

$u(\mathbf{x}[\{I_2 \cap I_1 \cap I_3\}]) - u(\mathbf{x}[\{I_1 \cap I_3\}]) = u(\mathbf{x}[\{2,4\}]) - u(\mathbf{x}[\{2\}]) + u(\mathbf{x}[\{2\}]) - u(\mathbf{x}[\{2\}]) = u(\mathbf{x}[\{2,4\}]) - u(\mathbf{x}[\{2\}])$.

The search algorithm is guaranteed to terminate, because the search follows edges from high-numbered nodes to lower-numbered nodes. For a given factor F_j, the running time is exponential in the length of the longest path in a GAI graph from j to some node $k < j$ that intersects j (since there might be exponentially many paths from j to k). In the worst case, when the intersection of I_j and all the preceding factors I_k ($k < j$) is non-empty, the number of relevant sets (or, alternatively, the number of paths from node $j-1$ to node 1), might be exponential in $j-1$. The total running time of the algorithm is exponential in the size of the largest set of factors with at least one common attribute. Let D be the largest intersection among all factors:

$$D = \max_{\substack{S \subseteq \{1,\ldots,M\}, \\ \cap_{s \in S} I_s \neq \emptyset}} |S|.$$

Then, the relevant set algorithm running time is exponential in D. If the factors are sparsely connected (D is bounded by a constant), or if the number of factors is not too large, finding the relevant sets is a computationally fast procedure.

Local configurations and basic outcomes

Knowing relevant sets enables us to represent GAI utilities in a convenient parameterized form that is decision-theoretically sound and is well-suited for elicitation. Before that, we need to introduce the equivalence between a factor's local configurations and its basic outcomes.

While $\mathbf{x}$ is some instantiation of all N attributes, $\mathbf{x}_j$ is a restricted instantiation of attributes in factor F_j only; we refer to $\mathbf{x}_j$ as a local suboutcome, or a *local configuration*. Let N_j be the number of all possible local configurations (settings of attributes) in factor F_j (e.g., with 3 boolean attributes, $N_j = 8$). We also assume that we can enumerate all local configurations in some consistent way, based on their attribute and attribute value indices. Then, if $1 \leq k \leq N_j$ is the index of some local configuration in factor F_j, we refer to that configuration as $\mathbf{x}_j^k$.

Conversely, $ind(\mathbf{x}_j)$ specifies the index of the local configuration $\mathbf{x}_j$ (so that $ind(\mathbf{x}_j^k) = k$).

Example 3.8 In factor F_2 with two binary attributes X_2 and X_3, the four local configurations are:

$$\mathbf{x}_2^1 = (x_2^0, x_3^0),$$
$$\mathbf{x}_2^2 = (x_2^1, x_3^0),$$
$$\mathbf{x}_2^3 = (x_2^0, x_3^1),$$
$$\mathbf{x}_2^4 = (x_2^1, x_3^1).$$

If an attribute X_i is binary, one of the values has to be the reference value (x_i^0); the other value is x_i^1. Recall that B_j is the set of basic outcomes of factor F_j, consisting of all possible outcomes $\mathbf{x}$ with attributes outside factor F_j fixed at reference levels: $B_j = \{\mathbf{x}[I_j], \ \ \mathbf{x} \in \mathbf{X}\}$. The size of B_j is N_j, the number of local configurations in factor F_j. This is because the attributes outside factor F_j are all fixed (at reference levels). Furthermore, there exists a one-to-one correspondence between local configurations and basic outcomes in each factor F_j. Each local configuration $\mathbf{x}_j^k$ can be *extended* to a full outcome by fixing the values of other attributes to their reference levels; we denote an extension of $\mathbf{x}_j^k$ as $ext(\mathbf{x}_j^k) = \mathbf{x}[I_j]^k = \mathbf{b}^{j,k}$. Thus, $\mathbf{b}^{j,k}$ is the k^{th} basic outcome of factor F_j. Every extension of a local configuration is a basic outcome for factor F_j:

$$\begin{aligned} B_j &= \{\mathbf{x}[I_j], \ \ \mathbf{x} \in \mathbf{X}\} \\ &= \{ext(\mathbf{x}_j^k) = \mathbf{x}[I_j]^k = \mathbf{b}^{j,k}, \ \ k = 1, \ldots, N_j\}. \end{aligned} \tag{3.13}$$

Example 3.9 Continuing Example 3.8, we can obtain the four basic outcomes corresponding to the four local configurations of factor F_2 by setting the attributes outside F_2 to reference

values:

$$\mathbf{x}_2^1 = (x_2^0, x_3^0) \longleftrightarrow \mathbf{b}^{2,1} = (x_1^0, x_2^0, x_3^0),$$
$$\mathbf{x}_2^2 = (x_2^1, x_3^0) \longleftrightarrow \mathbf{b}^{2,2} = (x_1^0, x_2^1, x_3^0),$$
$$\mathbf{x}_2^3 = (x_2^0, x_3^1) \longleftrightarrow \mathbf{b}^{2,3} = (x_1^0, x_2^0, x_3^1),$$
$$\mathbf{x}_2^4 = (x_2^1, x_3^1) \longleftrightarrow \mathbf{b}^{2,4} = (x_1^0, x_2^1, x_3^1).$$

Structure coefficients

We use the results of the GAI graph search algorithm (i.e., relevant sets L_j for each factor), and basic outcome sets B_j to define the final parameterized form for GAI utility functions. From Eq. 3.12, we can see that all outcomes on the right side are basic outcomes for factor F_j. Since the total number of basic outcomes (for factor F_j) is N_j, the number of local configurations, the subutility value for the r^{th} suboutcome of factor F_j can be written as:

$$\begin{aligned} u_j(\mathbf{x}_j^r) &= u(\mathbf{x}^r[I_j]) + \sum_{S \in L_j} (-1)^{|S|}\, u(\mathbf{x}^r[\bigcap_{s \in S} I_s \cap I_j]) + \sigma_{\bar{R}}\, u(\mathbf{x}^0) \\ &= \sum_{k=1}^{N_j} \mathbf{C}_j[r,k]\; u(\mathbf{b}^{j,k}) \\ &= \sum_{k=1}^{N_j} \mathbf{C}_j[r,k]\; \theta_j^k, \end{aligned} \tag{3.14}$$

where $\theta_j^k = u(\mathbf{b}^{j,k})$ is the utility of the k^{th} basic outcome of factor F_j, and $\mathbf{C}_j$ is a square $N_j \times N_j$ matrix of integer *structure coefficients*. The procedure for computing the structure coefficients (based on Eq. 3.14) is shown in Figure 3.5.

Thus, even though in Eq. 3.8 the number of addends is 2^{j-1}, since all of them are basic outcomes, the number of *unique* addends cannot exceed N_j (see Observation 3.4). Our GAI graph search procedure identifies the relevant ones, which allows us to compute the structure coefficients and express the GAI function as a linear combination of basic outcome utilities:

$$u(\mathbf{x}) = \sum_{j=1}^{M} \sum_{k=1}^{N_j} C_{\mathbf{x}_j}^k\; \theta_j^k, \tag{3.15}$$

Input: relevant sets L_j for each factor F_j
Output: for each factor F_j, a square $N_j \times N_j$ matrix $\mathbf{C}_j$ of structure coefficients

foreach *factor* F_j **do**
  $\mathbf{C}_j = \mathbf{I}$ (identity matrix)
  $\sigma_{\bar{R}} \leftarrow -1 - \sum_{S \in L_j} (-1)^{|S|}$ ($\sigma_{\bar{R}} \leftarrow 0$ if $F_j = F_1$)
  foreach *local configuration* $\mathbf{x}_j^r, r = 1, \ldots, N_j$ **do**
    foreach *set* S *in* L_j **do**
      $k \leftarrow ind(\mathbf{x}_j^r[\bigcap_{s \in S} I_s])$
      $\mathbf{C}_j[r,k] \leftarrow \mathbf{C}_j[r,k] + (-1)^{|S|}$
    end
    $\mathbf{C}_j[r,k_0] \leftarrow \mathbf{C}_j[r,k_0] + \sigma_{\bar{R}}$, where k_0 is the index of the reference configuration
  end
end

Figure 3.5: Procedure for computing structure coefficients $\mathbf{C}_j$

where $C^k_{\mathbf{x}_j} = \mathbf{C}_j[ind(\mathbf{x}_j), k]$. If we knew all parameters θ_j^k, we could find the utility of any outcome $\mathbf{x} \in \mathbf{X}$. Since structure coefficients are fixed given the GAI factor decomposition, the goal of our elicitation procedures will be to estimate the GAI parameters θ_j^k.

The structure matrix $\mathbf{C}_j$ for each factor F_j consists of $|\mathbf{X}_j|^2$ entries. The number of all GAI parameters θ_j^k is the same as the number of all factor configurations:

$$|\{\theta_1^1, \ldots, \theta_j^k, \ldots, \theta_M^{N_M}\}| = \sum_{j=1}^{M} |\mathbf{X}_j|.$$

Thus, apart from structure matrices $\mathbf{C}_j$, by using the GAI representation in Eq. 3.15, we do not increase the size of GAI representation (compared to the traditional factored GAI representation).

Example 3.10 We continue the previous examples with $I_1 = \{1,2\}$ and $I_2 = \{2,3\}$, and all binary attributes. From the dependency structure, we know that

$$u_2(x_2, x_3) = u(\mathbf{x}[I_2]) - u(\mathbf{x}([I_1 \cap I_2]) = u(x_1^0, x_2, x_3) - u(x_1^0, x_2, x_3^0),$$

where x_i is a generic binary attribute that could take the values x_i^0 or x_i^1. Then,

$$u_2(\mathbf{x}_2^1) = u_2(x_2^0, x_3^0) = u(x_1^0, x_2^0, x_3^0) - u(x_1^0, x_2^0, x_3^0) = 0\,\theta_2^1 + 0\,\theta_2^2 + 0\,\theta_2^3 + 0\,\theta_2^4,$$
$$u_2(\mathbf{x}_2^2) = u_2(x_2^1, x_3^0) = u(x_1^0, x_2^1, x_3^0) - u(x_1^0, x_2^1, x_3^0) = 0\,\theta_2^1 + 0\,\theta_2^2 + 0\,\theta_2^3 + 0\,\theta_2^4,$$
$$u_2(\mathbf{x}_2^3) = u_2(x_2^0, x_3^1) = u(x_1^0, x_2^0, x_3^1) - u(x_1^0, x_2^0, x_3^0) = -1\,\theta_2^1 + 0\,\theta_2^2 + 1\,\theta_2^3 + 0\,\theta_2^4,$$
$$u_2(\mathbf{x}_2^4) = u_2(x_2^1, x_3^1) = u(x_1^0, x_2^1, x_3^1) - u(x_1^0, x_2^1, x_3^0) = 0\,\theta_2^1 - 1\,\theta_2^2 + 0\,\theta_2^3 + 1\,\theta_2^4.$$

The coefficients $C_{\mathbf{x}_j}$ specify which basic outcomes are involved in defining the subutility values. In this example, $\mathbf{C}_{\mathbf{x}_2^3} = \mathbf{C}[3,:] = [-1, 0, 1, 0]$.

We can notice that in the example above, the sum of the coefficient vector values is zero (except for factor F_1, when the entries sum up to 1). In fact, it is a general result.

Observation 3.6 *The rows of coefficient matrices $\mathbf{C}_j$ sum up to 0, for $j \geq 2$:*

$$\sum_k \mathbf{C}_j[r,k] = 0, \textit{ for all } j \geq 2, \textit{ and all indices } 1 \leq r \leq N_j.$$

The coefficient matrix $\mathbf{C}_1$ is an identity matrix.

Proof This result follows directly from Eq. 3.14 and Observation 3.3. For $j \geq 2$, the sum of all coefficients in front of the basic outcomes on the first line of Eq. 3.14 is 0 (see Observation 3.3). Therefore, the sum of all coefficients on line 3 (and 4) of Eq. 3.14 must also be 0. When $j = 1$, $u_1(\mathbf{x}_1^r) = u(\mathbf{b}^{1,r}) = \theta_1^r$. Therefore, $\mathbf{C}_1[r,r] = 1$ for all indices r, while all other entries are 0. Thus, $\mathbf{C}_1$ is an identity matrix. □

3.2.3 Local value functions

Additive models

Additive utilities facilitate not only representation, but also elicitation of user preferences, partly because utility function parameters can be assessed using almost exclusively local queries. Local queries ask a user for the strength of preference over each attribute in isolation; the values

of other attributes are, because of independence, irrelevant. As we saw in Section 2.1.3, any additive utility function can be written in a form where an attribute subutility u_i is a product of *local value functions (LVFs)* v_i (normalized to the [0,1] interval) and scaling constants λ_i (all positive, and summing up to 1) (Eq. 2.15):

$$u(\mathbf{x}) = \sum_{i=1}^{N} u_i(x_i) = \sum_{i=1}^{N} \lambda_i v_i(x_i).$$

This simple factorization separates the representation of preferences into two components: "local" and "global." Since we can define LVFs independently of other attributes, we can also *assess* them independently using local queries involving only the attribute in question. LVFs can be defined using local lotteries that involve only a single attribute: $v_i(x_i) = p$, where p is the probability at which the user is indifferent between two local outcomes x_i and $\langle p, x_i^{\top}; 1-p, x_i^{\perp}\rangle$, *ceteris paribus*.

Because of the interdependence of the values of local configurations across different factors, the separation of preference representation into local and global components for GAI models is more involved. Below, we demonstrate how we can confer the benefits of local structure upon the more expressive GAI utility models.

Local values in GAI models

The parameterized GAI representation $u(\mathbf{x}) = \sum_{j=1}^{M} \sum_{k=1}^{N_j} C^k_{\mathbf{x}_j} \theta^k_j$ has many advantages. The structure coefficients can be computed if the GAI factor decomposition is known, and the parameters θ^k_j have a semantically sound interpretation: they represent utilities of basic outcomes $\mathbf{b}^{j,k}$. However, basic outcomes are global outcomes, and therefore θ^k_j are global parameters. As with additive utilities, we would like to exploit the local preference structure to facilitate elicitation. Our goal is to extend the local value functions to GAI utilities.

First, a few more definitions. Let $\mathbf{b}^{j,\top}$ be the best outcome in B_j and $\mathbf{b}^{j,\perp}$ be the worst outcome in B_j. We will refer to them as *top* and *bottom anchors* for factor F_j.

Definition 3.4 (Top and bottom anchors) A global basic outcome $\mathbf{b}^{j,\top}$ is the *top anchor* for factor F_j if $\mathbf{b}^{j,\top} \succeq \mathbf{x}[I_j],\ \forall \mathbf{x} \in \mathbf{X}$. A global basic outcome $\mathbf{b}^{j,\bot}$ is the *bottom anchor* for factor F_j if $\mathbf{b}^{j,\bot} \preceq \mathbf{x}[I_j],\ \forall \mathbf{x} \in \mathbf{X}$.

We denote the utility of the top anchor as $\theta_j^\top = u(\mathbf{b}^{j,\top})$, and utility of the bottom anchor as $\theta_j^\bot = u(\mathbf{b}^{j,\bot})$. Then, by using the expected utility theorem 2.8, the utility of any basic outcome in B_j can be written as an affine combination of anchor utilities:

$$\begin{gathered}\theta_j^k = p_j^k\, \theta_j^\top + (1 - p_j^k)\, \theta_j^\bot \\ \Longleftrightarrow \\ \mathbf{b}^{j,k} \sim \langle p_j^k, \mathbf{b}^{j,\top};\ 1 - p_j^k, \mathbf{b}^{j,\bot} \rangle. \end{gathered} \tag{3.16}$$

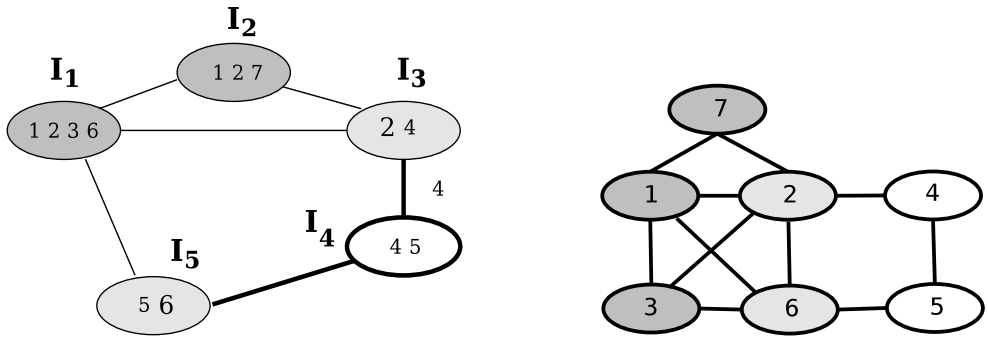
Here, p_j^k is the probability at which one would be indifferent between the outcome $\mathbf{b}^{j,k}$ and the lottery in which the top anchor $\mathbf{b}^{j,\top}$ is realized with probability p_j^k, and the bottom anchor $\mathbf{b}^{j,\bot}$ is realized with probability $1 - p_j^k$.

By plugging $\theta_j^k = p_j^k\, \theta_j^\top + (1 - p_j^k)\, \theta_j^\bot$ into Eq. 3.14, we get

$$\begin{aligned} u_j(\mathbf{x}_j^r) &= \sum_{k=1}^{N_j} \mathbf{C}_j[r,k]\ \theta_j^k \\ &= \sum_{k=1}^{N_j} \mathbf{C}_j[r,k]\ (p_j^k\, \theta_j^\top + (1 - p_j^k)\, \theta_j^\bot) \\ &= \sum_{k=1}^{N_j} \mathbf{C}_j[r,k]\ p_j^k\, \theta_j^\top + \sum_{k=1}^{N_j} \mathbf{C}_j[r,k]\ \theta_j^\bot - \sum_{k=1}^{N_j} \mathbf{C}_j[r,k]\ p_j^k\, \theta_j^\bot \\ &= (\theta_j^\top - \theta_j^\bot) \sum_{k=1}^{N_j} \mathbf{C}_j[r,k]\ p_j^k + \theta_j^\bot \sum_{k=1}^{N_j} \mathbf{C}_j[r,k]. \end{aligned}$$

In the previous Observation 3.6, we showed that $\sum_{k=1}^{N_j} \mathbf{C}_j[r,k]$ is always 0, unless $j = 1$, in which case it is 1. Therefore, we can rewrite the GAI subutility equations as follows:

$$\begin{aligned} u_1(\mathbf{x}_1^r) &= (\theta_1^\top - \theta_1^\bot)\ p_j^r + \theta_1^\bot, \\ u_j(\mathbf{x}_j^r) &= (\theta_j^\top - \theta_j^\bot)\ \sum_{k=1}^{N_j} \mathbf{C}_j[r,k]\ p_j^k. \end{aligned} \tag{3.17}$$



(a) GAI graph (b) CA-independence graph

Figure 3.6: Conditioning set $\{X_2, X_6\}$ for factor F_4 in a GAI graph (a) and a CA-independence graph (Bacchus and Grove, 1995) (b).

We will later show that the probabilities p_j^k have a local interpretation, which will enable us to define LVFs for GAI models. But first, we introduce factor conditioning sets.

Factor conditioning sets

For each factor, there exists a set of attributes (outside that factor) that "shield" the influence of all other attributes in the domain. We call that set of attributes the *conditioning set*. A related concept in the factored probability models is known as the *Markov blanket* (Pearl, 1988).

Definition 3.5 The *conditioning set* K_j for factor F_j is the set of all attributes that share GAI factors with attributes in F_j:

$$K_j = (\bigcup_{k:F_k \cap F_j \neq \emptyset} F_k) \setminus F_j.$$

The set of indices of the attributes in K_j will be denoted as I_{K_j}.

Example 3.11 Let $u(x_1, \ldots, x_7) = u_1(x_1, x_2, x_3, x_6) + u_2(x_1, x_2, x_7) + u_3(x_2, x_4) + u_4(x_4, x_5) + u_5(x_5, x_6)$ (see Fig. 3.6). Then, the conditioning set for F_4 is $K_4 = \{X_2, X_6\}$ (and $I_{K_4} = \{2, 6\}$), since F_4 is connected to F_3 and F_5, and X_2, X_6 are all the attributes in F_3 and F_5 that are not in F_4 (Fig. 3.6a). Alternatively, we can consider a graph where nodes are individual attributes, and there is an edge between two attributes if and only if they are together in some factor. Such a graph is called the *CA-independence map* (where CA stands for "conditionally additive") by Bacchus and Grove (1995). In this graph, the conditioning set for the factor F_i

is the smallest set of nodes that block all paths between the factor F_i and other nodes (see Fig. 3.6b). The GAI graph can be viewed as a clique graph of the CA-independence map.

GAI local value theorem

A conditioning set separates a GAI factor from all other attributes and allows us to define semantically valid local preference relations over factor instantiations.

Theorem 3.2 (GAI local value theorem)

Let $F_1, \ldots, F_M$ be the factors of a GAI decomposition of some utility function (i.e., $F_1, \ldots, F_M$ satisfy the GAI condition). For any $j \in \{1, \ldots, M\}$, after an appropriate rearrangement of indices, any outcome $\mathbf{x} \in \mathbf{X}$ can be written as $(\mathbf{x}_j, \mathbf{z}_j, \mathbf{y})$, where $\mathbf{x}_j \in \mathbf{X}_j$ instantiates attributes in factor F_j, $\mathbf{z}_j \in \mathbf{X}_{I_{K_j}}$ is a suboutcome in its conditioning set, and $\mathbf{y}$ is an instantiation of all remaining attributes. Then,

$$(\mathbf{x}_j, \mathbf{z}_j, \mathbf{y}) \sim \langle p, (\mathbf{x}_j^\top, \mathbf{z}_j, \mathbf{y});\ 1-p, (\mathbf{x}_j^\perp, \mathbf{z}_j, \mathbf{y})\rangle \implies$$
$$(\mathbf{x}_j, \mathbf{z}_j, \mathbf{y}') \sim \langle p, (\mathbf{x}_j^\top, \mathbf{z}_j, \mathbf{y}');\ 1-p, (\mathbf{x}_j^\perp, \mathbf{z}_j, \mathbf{y}')\rangle, \text{ for any } \mathbf{y}' \in \mathbf{X}_{(I_j \cup I_{K_j})^C}.$$

Therefore, by definition of a local lottery,

$$(\mathbf{x}_j, \mathbf{z}_j) \sim \langle p, (\mathbf{x}_j^\top, \mathbf{z}_j);\ 1-p, (\mathbf{x}_j^\perp, \mathbf{z}_j)\rangle.$$

As long as the attributes in the conditioning set K_j are fixed, no other attributes influence the strength of preference for local outcomes in factor F_j. We can therefore perform *local* elicitation with respect to local anchors $\mathbf{x}_j^\top$ and $\mathbf{x}_j^\perp$, without specifying the levels of the complementary attributes $\mathbf{y}$.

Proof

By the expected utility theorem,

$$(\mathbf{x}_j, \mathbf{z}_j, \mathbf{y}) \sim \langle p, (\mathbf{x}_j^\top, \mathbf{z}_j, \mathbf{y});\ 1-p, (\mathbf{x}_j^\perp, \mathbf{z}_j, \mathbf{y})\rangle \implies$$
$$u(\mathbf{x}_j, \mathbf{z}_j, \mathbf{y}) = p\, u(\mathbf{x}_j^\top, \mathbf{z}_j, \mathbf{y}) + (1-p)\, u(\mathbf{x}_j^\perp, \mathbf{z}_j, \mathbf{y}).$$

We partition the set of all attributes $\{X_1, \ldots, X_N\}$ into the factor F_j, its conditioning set K_j, and remaining attributes $Y_j = (F_j \cup K_j)^C$. The corresponding attribute index sets are I_j, I_{K_j}, and I_{Y_j}.

The set $T_j = \{k : F_k \cap F_j \neq \emptyset\} \backslash \{j\}$ contains indices of all factors that share some attributes with the factor F_j (in a GAI graph, it is the set of all nodes connected to node j). By definition, the factors indexed by T_j contain only the attributes in F_j and K_j (because $K_j = \bigcup_{k \in T_j} F_k$). All other factors F_l, with $l \notin T_j \cup \{j\}$, contain only atributes in K_j and Y_j (F_l cannot have attributes in F_j because then l would be in T_j, which is a contradiction).

Since u is a GAI function,

$$\begin{aligned} u(\mathbf{x}_j, \mathbf{z}_j, \mathbf{y}) &= \left(u_j(\mathbf{x}_j) + \sum_{k \in T_j} u_k(\mathbf{x}_k) \right) + \sum_{l \notin T_j \cup \{j\}} u_l(\mathbf{x}_l) \\ &= u_X(\mathbf{x}_{I_j \cup I_{K_j}}) + u_Y(\mathbf{x}_{I_{K_j} \cup I_{Y_j}}) \\ &= u_X(\mathbf{x}_j, \mathbf{z}_j) + u_Y(\mathbf{z}_j, \mathbf{y}), \end{aligned}$$

since $\mathbf{x}_j \in \mathbf{X}_j$, $\mathbf{z}_j \in \mathbf{X}_{I_{K_j}}$, and $\mathbf{y} \in \mathbf{X}_{I_{Y_j}}$. The domain of the subutility function u_X comprises all the attributes in F_j and its conditioning set K_j, whereas the domain of u_Y consists of attributes in K_j and Y_j. Therefore, the following holds:

$$\begin{aligned} u(\mathbf{x}_j, \mathbf{z}_j, \mathbf{y}) = p\, u(\mathbf{x}_j^\top, \mathbf{z}_j, \mathbf{y}) + (1-p)\, u(\mathbf{x}_j^\perp, \mathbf{z}_j, \mathbf{y}) &\iff \\ u_X(\mathbf{x}_j, \mathbf{z}_j) + u_Y(\mathbf{z}_j, \mathbf{y}) = p\,(u_X(\mathbf{x}_j^\top, \mathbf{z}_j) + u_Y(\mathbf{z}_j, \mathbf{y})) + (1-p)\,(u_X(\mathbf{x}_j^\perp, \mathbf{z}_j) + u_Y(\mathbf{z}_j, \mathbf{y})) &\iff \\ u_X(\mathbf{x}_j, \mathbf{z}_j) = p\, u_X(\mathbf{x}_j^\top, \mathbf{z}_j) + (1-p)\, u_X(\mathbf{x}_j^\perp, \mathbf{z}_j) &\iff \\ (\mathbf{x}_j, \mathbf{z}_j) \sim \langle p, (\mathbf{x}_j^\top, \mathbf{z}_j);\ 1-p, (\mathbf{x}_j^\perp, \mathbf{z}_j) \rangle. \quad \square \end{aligned}$$

Local value functions

Theorem 3.2 tells us that the utility of any suboutcome $\mathbf{x}_j$ (in factor F_j) can be expressed locally in terms of the two anchor levels, assuming fixed values $\mathbf{z}_j \in K_j$ of attributes in the conditioning set: $(\mathbf{x}_j, \mathbf{z}_j) \sim \langle p, (\mathbf{x}_j^\top, \mathbf{z}_j);\ 1-p, (\mathbf{x}_j^\perp, \mathbf{z}_j) \rangle$. This holds for any instantiation of

the conditioning set. Assuming a *fixed* instantiation of the conditioning set, we can now define a local value function for GAI utilities.

Definition 3.6 (Local value function) A *local value function* (LVF) v_j is a function on $\mathbf{X}_j$ such that

$$v_j(\mathbf{x}_j) = p$$

if and only if

$$(\mathbf{x}_j, \mathbf{z}_j^0) \sim \langle p, (\mathbf{x}_j^\top, \mathbf{z}_j^0);\ 1-p, (\mathbf{x}_j^\perp, \mathbf{z}_j^0)\rangle,$$

for all $\mathbf{x}_j \in \mathbf{X}_j$, and all attributes in the conditioning set for factor F_j *fixed at the reference level* $\mathbf{z}_j^0 \in \mathbf{X}_{I_{K_j}}$. The range of the LVF v_j is constrained to the $[0,1]$ interval, with $v_j(\mathbf{x}_j^\top) = 1$, and $v_i(\mathbf{x}_j^\perp) = 0$.

GAI representation using local value functions

From the definition of the local value function, we can see that $v_j(\mathbf{x}_j^k) = p_j^k$ implies $(\mathbf{x}_j^k, \mathbf{z}_j^0) \sim \langle p_j^k, (\mathbf{x}_j^\top, \mathbf{z}_j^0);\ 1-p_j^k, (\mathbf{x}_j^\perp, \mathbf{z}_j^0)\rangle$. By definition of the local lottery, $v_j(\mathbf{x}_j^k) = p_j^k$ also implies $(\mathbf{x}_j^k, \mathbf{z}_j^0, \mathbf{y}^0) \sim \langle p_j^k, (\mathbf{x}_j^\top, \mathbf{z}_j^0, \mathbf{y}^0);\ 1-p_j^k, (\mathbf{x}_j^\perp, \mathbf{z}_j^0, \mathbf{y}^0)\rangle$, where we chose to set the complementary attributes to $\mathbf{y}^0$. The outcomes $(\mathbf{x}_j^\top, \mathbf{z}_j^0, \mathbf{y}^0) = \mathbf{b}^{j,\top}$ and $(\mathbf{x}_j^\perp, \mathbf{z}_j^0, \mathbf{y}^0) = \mathbf{b}^{j,\perp}$ are the two anchor outcomes of factor F_j, while the outcome $(\mathbf{x}_j^k, \mathbf{z}_j^0, \mathbf{y}^0) = \mathbf{b}^{j,k}$ is the k^{th} basic outcome of factor F_j. Thus, the local value function $v_j(\mathbf{x}_j^k)$ calibrates the relative utility of $\mathbf{b}^{j,k}$ with respect to the two anchor outcomes (see Eq. 3.16):

$$\begin{aligned}
&\mathbf{b}^{j,k} \sim \langle p_j^k, \mathbf{b}^{j,\top};\ 1-p_j^k, \mathbf{b}^{j,\perp}\rangle \\
&\qquad\Longleftrightarrow \\
&\mathbf{b}^{j,k} \sim \langle v_j(\mathbf{x}_j^k), \mathbf{b}^{j,\top};\ 1-v_j(\mathbf{x}_j^k), \mathbf{b}^{j,\perp}\rangle \\
&\qquad\Longleftrightarrow \\
&\theta_j^k = v_j(\mathbf{x}_j^k)\,\theta_j^\top + (1-v_j(\mathbf{x}_j^k))\,\theta_j^\perp, \qquad (3.18)
\end{aligned}$$

where, as before, $\theta_j^k = u(\mathbf{b}^{j,k})$ denotes the utility of the k^{th} basic outcome in factor F_j. Rewriting Eq. 3.17, we get:

$$\begin{aligned} u_1(\mathbf{x}_1^r) &= [u(\mathbf{b}^{1,\top}) - u(\mathbf{b}^{1,\perp})]\ v_j(\mathbf{x}_j^r) + u(\mathbf{b}^{1,\perp}), \\ u_j(\mathbf{x}_j^r) &= [u(\mathbf{b}^{j,\top}) - u(\mathbf{b}^{j,\perp})]\ \sum_{k=1}^{N_j} \mathbf{C}_j[r,k]\ v_j(\mathbf{x}_j^k), \end{aligned} \tag{3.19}$$

and, therefore,

$$u(\mathbf{x}^r) = \theta_1^\perp + \sum_{j=1}^{M} \left[[u(\mathbf{b}^{j,\top}) - u(\mathbf{b}^{j,\perp})] \sum_{k=1}^{N_j} \mathbf{C}_j[r,k]\ v_j(\mathbf{x}_j^k) \right].$$

Since $\theta_1^\perp$ is a constant, we can use a simpler equivalent utility:

$$u(\mathbf{x}^r) = \sum_{j=1}^{M} \left[[u(\mathbf{b}^{j,\top}) - u(\mathbf{b}^{j,\perp})] \sum_{k=1}^{N_j} \mathbf{C}_j[r,k]\ v_j(\mathbf{x}_j^k) \right]. \tag{3.20}$$

By contrasting this local GAI parameterization with the previous global parameterization (Eq. 3.15)

$$u(\mathbf{x}^r) = \sum_{j=1}^{M} \sum_{k=1}^{N_j} \mathbf{C}_j[r,k]\ u(\mathbf{b}^{j,k}),$$

we can observe that the global representation requires knowledge of utilities of *all* basic outcomes $\mathbf{b}^{j,k}$ (which are global outcomes), while the local representation depends only on local value functions of local configurations $\mathbf{x}_j^k$ and a much smaller set of utilities of global outcomes $\mathbf{b}^{j,\top}$ and $\mathbf{b}^{j,\perp}$ (viz., the $2M$ anchor utilities).

Both the local and global parameterizations of GAI utilities are equivalent. The local parameterization (Eq. 3.20) shows that a GAI model, similar to simple additive utility functions, can be additively decomposed into factors which are the product of global parameters (weights) $\lambda_j = u(\mathbf{b}^{j,\top}) - u(\mathbf{b}^{j,\perp})$ and a linear combination of LVF parameters $v_j(\mathbf{x}_j^k)$:

$$u(\mathbf{x}^r) = \sum_{j=1}^{M} \left[\lambda_j \sum_{k=1}^{N_j} \mathbf{C}_j[r,k]\ v_j(\mathbf{x}_j^k) \right]. \tag{3.21}$$

Example 3.12 We illustrate the two GAI representations with the simple non-trivial two-factor, three-attribute GAI model with $I_1 = \{1, 2\}$ and $I_2 = \{2, 3\}$:

$$\begin{aligned} u(x, y, z) =& u(x, y, z^0) + (u(x^0, y, z) - u(x^0, y, z^0)) \\ =& [u(x^\top, y^\top, z^0) - u(x^\perp, y^\perp, z^0)]\ v_1(x, y) + \\ & [u(x^0, y^\top, z^\top) - u(x^0, y^\perp, z^\perp)]\ (v_2(y, z) - v_2(y, z^0)). \\ =& \lambda_1\ v_1(x, y) + \lambda_2\ (v_2(y, z) - v_2(y, z^0)). \end{aligned}$$

Relation to additive utilities

GAI local value functions generalize the concept of local value functions for additive utilities. Let's consider the GAI expression (Eq. 3.20):

$$u(\mathbf{x}^r) = \sum_{j=1}^{M} \left[[u(\mathbf{b}^{j,\top}) - u(\mathbf{b}^{j,\perp})] \sum_{k=1}^{N_j} \mathbf{C}_j[r, k]\ v_j(\mathbf{x}_j^k) \right].$$

In the additive utility case, with no overlapping factors, $\mathbf{C}_j[r, k]$ = 1 if $r = k$, and 0 otherwise (for all factors, or attributes, F_j). Furthermore, following a common practice for additive utilities, let's assume that the reference outcome is the worst possible outcome, i.e., $\mathbf{x}^0 = \mathbf{x}^\perp$, and utility function u is normalized to the $[0, 1]$ interval. Therefore, $\mathbf{b}^{j,\perp} = \mathbf{x}^\perp$ and $u(\mathbf{b}^{j,\perp}) = 0$ for all j:

$$u(\mathbf{x}^r) = \sum_{j=1}^{M} u(\mathbf{b}^{j,\top}) v_j(\mathbf{x}_j^r) = \sum_{j=1}^{M} \lambda_j v_j(\mathbf{x}_j^r),$$

which is a familiar expression for additive utilites (see Eq. 2.21).

3.2.4 Summary

In the previous sections, we have introduced two new GAI model representations. The first one (Eq. 3.15) defines the utility of any outcome $\mathbf{x} \in \mathbf{X}$ as linear combination of basic outcome utilities θ_j^k:

$$u(\mathbf{x}) = \sum_{j=1}^{M} \sum_{k=1}^{N_j} C_{\mathbf{x}_j}^k\ \theta_j^k \tag{3.22}$$

($\theta_j^k = u(\mathbf{b}^{j,k})$ is a parameter defined as the utility of the basic outcome $\mathbf{b}^{j,k}$). Such parameters are global, since we need to know utilities of global basic outcomes to assess them.

The second representation uses mostly local LVF parameters $v_j^k = v_j(\mathbf{x}_j^k)$, except for the $2M$ global parameters $\theta_1^\top, \theta_1^\perp, \ldots, \theta_M^\top, \theta_M^\perp$ (Eq. 3.20):

$$u(\mathbf{x}) = \sum_{j=1}^{M} \left[[\theta_j^\top - \theta_j^\perp] \sum_{k=1}^{N_j} C_{\mathbf{x}_j}^k \; v_j^k \right]. \tag{3.23}$$

LVF parameters are local, since they are defined as the LVFs of local factor outcomes. To assess them, we only need to consider attributes in the relevant factor and its conditioning set.

Since structure coefficients $\mathbf{C}_j$ are fixed given the GAI factor decomposition, a GAI utility function is fully determined by either (a) the global parameters $\{\theta_1^1, \ldots, \theta_j^k, \ldots, \theta_M^{N_M}\}$; or (b) $2M$ global parameters $\{\theta_1^\top, \theta_1^\perp, \ldots, \theta_M^\top, \theta_M^\perp\}$ and the local LVF parameters $\{v_1^1, \ldots, v_j^k, \ldots, v_M^{N_M}\}$. In both representations, the number of parameters is equal to the number of all factor configurations:

$$|\{\theta_1^1, \ldots, \theta_j^k, \ldots, \theta_M^{N_M}\}| = |\{v_1^1, \ldots, v_j^k, \ldots, v_M^{N_M}\}| = \sum_{j=1}^{M} |\mathbf{X}_j|.$$

Global GAI parameters and local LVF parameters are related:

$$\theta_j^k = v_j^k \, \theta_j^\top + (1 - v_j^k) \, \theta_j^\perp.$$

Therefore, information about local parameters can be used to assess global parameters as well. In the following section, we describe several practical query types that provide information about local and global GAI utility parameters.

3.3 Queries for eliciting GAI model parameters

Previous elicitation procedures for GAI models relied on full outcome queries which circumvent the problem of local elicitation and global calibration issues. Such semantically sound procedures were implicitly described by Fishburn (1967b); more recently, Gonzales and Perny (2004) suggested an explicit elicitation strategy that uses direct global queries. However, by

resorting to full outcome queries, we lose some of the advantages of additive models and fail to exploit the decomposition of utility functions *during the elicitation process*.

In interactive elicitation settings, the types of queries we pose to the user impacts not only the difficulty or accuracy of user responses, but also directly relates to the modeling and computational complexity of a decision problem. Each response to a query adds an additional constraint to the set of constraints that define the feasible utility space. The shape and structure of that space in turn influences how hard it is to compute an optimal decision or a good elicitation policy. Queries that exploit structure in user preferences also make it easier to define tractable decision and elicitation models. In multiattribute settings, *local* queries only involve a (usually small) subset of all attributes; the values of remaining attributes do not matter, as long as they stay the same. In contrast, queries that ask a a user to consider (usually, compare or evaluate) full outcomes, are *global* queries. Global queries, if involving more than a few attributes, are much harder to assess. In general, we would like to minimize the number of arbitrary global queries in an elicitation process.

This section introduces several types of GAI queries that can be employed during *elicitation* of GAI utility parameters in both Bayesian and strict uncertainty settings. We focus on queries that are well-defined semantically, easy to explain to non-expert users and relatively simple to answer. We omit *direct* queries that ask the user for the exact utility of some (local or global) outcome,[1] since they are in most cases too difficult for a user to answer accurately (Keeney and Raiffa, 1976), and concentrate instead on cognitively simpler *bound* and *comparison* queries. Queries are further distinguished by the type of outcomes involved (*local* or *global*).

3.3.1 Local queries

Local queries involve only a small subset of all attributes, namely attributes in some GAI factor and the attributes in the factor's conditioning set; the values of the remaining attributes do not

[1] Such a query could also be recast in a form that asks about the probability p at which the user would be indifferent between some outcome $\mathbf{x}$ and the lottery $\langle p, \mathbf{x}^{\top};\ 1-p, \mathbf{x}^{\perp}\rangle$.

have to be taken into consideration. We use two types of local queries: comparison and sorting queries, and bound queries.

3.3.1.1 Local comparison queries (LCQs)

A *comparison query* asks the user to compare two outcomes and select the more preferred one. In a *local comparison query* (LCQ), both outcomes are local outcomes and belong to the same factor; in addition, the user is asked to assume that the attributes in the factor's conditioning set are fixed at reference levels. A sample query would be: "Assume that the attributes in K_j are fixed at reference levels. Would you prefer the local outcome $\mathbf{x}_j^i$ to the local outcome $\mathbf{x}_j^k$, ceteris paribus?" If the answer is "yes", then $v_j(\mathbf{x}_j^i) \geq v_j(\mathbf{x}_j^k)$; if "no", then $v_j(\mathbf{x}_j^i) < v_j(\mathbf{x}_j^k)$.

Responses to LCQs impose linear constraints on the GAI parameters θ_j^k from our standard GAI model $u(\mathbf{x}) = \sum_{j=1}^{M} \sum_{k=1}^{N_j} C_{\mathbf{x}_j}^k \, \theta_j^k$ (Eq. 3.15). To see this, first, from Eq. 3.18, we derive a relationship between LVFs and GAI parameters:

$$\begin{aligned} & \theta_j^k = v_j(\mathbf{x}_j^k)\, \theta_j^{\top} + (1 - v_j(\mathbf{x}_j^k))\, \theta_j^{\perp} \\ & \iff \\ & v_j(\mathbf{x}_j^k) = \frac{\theta_j^k - \theta_j^{\perp}}{\theta_j^{\top} - \theta_j^{\perp}}. \end{aligned} \tag{3.24}$$

A "yes" response leads to:

$$v_j(\mathbf{x}_j^i) \geq v_j(\mathbf{x}_j^k) \iff \frac{\theta_j^i - \theta_j^{\perp}}{\theta_j^{\top} - \theta_j^{\perp}} \geq \frac{\theta_j^k - \theta_j^{\perp}}{\theta_j^{\top} - \theta_j^{\perp}} \iff \theta_j^i \geq \theta_j^k.$$

Similarly, a "no" response imposes the constraint $\theta_j^i < \theta_j^k$.

Fig. 3.7 shows an example of the local comparison query used in the UTPREF recommendation system (which will be discussed in detail in Chapter 6).

3.3.1.2 Local sorting

Instead of asking a user to consider only two outcomes, we can present an interface which allows one to sort a list of outcomes in order of decreasing (or increasing) preference. *Sorting*

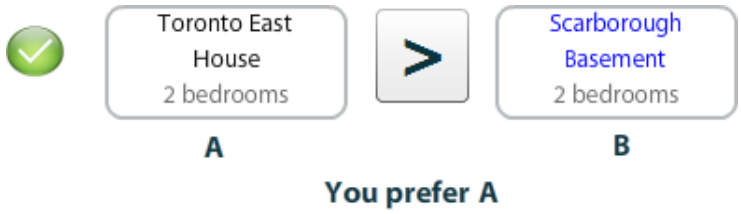


Figure 3.7: A local comparison query used in the UTPREF recommendation system. The outcomes in this domain are rental apartments in Toronto, defined by ten attributes, such as rental price, building type, number of bedrooms, and others. In this local comparison query, a user is asked to consider two local outcomes and select the better option. In this case, the two local outcomes are `Toronto East`, `House` and `Scarborough`, `Basement`, defined by the instantiation of *Area* and *Building type* attributes in the first factor. The conditioning set of the first factor consists of the *Number of bedrooms* attribute, which is set to the reference value `2 bedrooms`, distinguished by the grey font color. In an LCQ query, the user does not have to consider the values of other attributes.

Figure 3.8: A local sorting interface used in the UTPREF recommendation system. All local outcomes belong to a factor with *Housing type* and *Number of bedrooms* attributes. Notice that the *Area* attribute, which is the local conditioning set, is fixed at the same reference value for all outcomes (users are alerted to this fact elsewhere on the screen).

Figure 3.9: A local bound query used in the UTPREF recommendation system. The scale is 0 to 100. Here, the user has indicated that the value of the `Toronto East, House` outcome is somewhere between 0 and 50 by dragging the outcome in question to the lower bin. The conditioning set attribute *Number of bedrooms* is set to its reference value `2 bedrooms` and is distinguished by the grey font color. The user can also adjust the query "boundary" (in this case 50) by moving the slider to provide tighter or looser constraints if they feel comfortable doing so.

a list of factor outcomes determines all pairwise relationships between factor outcomes (i.e., it is equivalent to asking a large set of pairwise LCQs). In addition to the list of local outcomes, the user is asked to assume that the attributes in the conditioning set are fixed at reference levels. If the number of items in the list is L, the resulting order imposes $O(L^2)$ pairwise comparison constraints on the GAI parameters.

Fig. 3.8 shows an example of the local sorting interface used in the UTPREF system.

3.3.1.3 Local bound queries (LBQs)

A *bound query* asks the user to consider a single outcome, and decide whether its value is greater or less than some specified bound b. In a *local bound query* (LBQ), the outcome is a local factor outcome $\mathbf{x}_j^i$ whose LVF $v_j(\mathbf{x}_j^i)$ is between 0 and 1 (where 0 is the LVF value of the worst factor outcome $\mathbf{x}_j^\perp$ and 1 is the value of the best factor outcome $\mathbf{x}_j^\top$). One way of asking LBQs relies on the standard lottery semantics to calibrate the value of $\mathbf{x}_j^i$ with respect to $\mathbf{x}_j^\perp$ and $\mathbf{x}_j^\top$: "Assume that the attributes in K_j are fixed at reference levels. Would you prefer the local outcome $\mathbf{x}_j^i$ to a lottery $\langle b, \mathbf{x}_j^\top; 1-b, \mathbf{x}_j^\perp \rangle$, ceteris paribus?" If the answer is "yes", $v_j(\mathbf{x}_j^i) \geq b$; if "no", then $v_j(\mathbf{x}_j^i) < b$. The bound b is a probability. Such a binary (yes/no) LBQ differs from a direct local standard gamble query since we do not ask the user to *choose* the indifference level b, only to bound it.

A practical approximation of the probabilistic semantics can be achieved by asking the user to simply consider the outcome $\mathbf{x}_j^i$ on a scale from 0 to 1 (or 0 to 100), where 0 corresponds to the worst local outcome $\mathbf{x}_j^\perp$ and 1 corresponds to the best outcome $\mathbf{x}_j^\top$, and specify if the value of $\mathbf{x}_j^i$ is greater or less than the bound b (of course, as in other local queries, the attributes in the conditioning set are assumed fixed at reference values). Such non-probabilistic LBQs are arguably easier to explain and possibly easier to answer. We use them in the UTPREF system (see Fig. 3.9 for a sample screenshot), and evaluate them in more detail in Chapter 6.

A response to an LBQ imposes a linear constraint that ties together three different GAI utility parameters:

$$\begin{aligned} v_j(\mathbf{x}_j^i) \geq b &\iff \frac{\theta_j^i - \theta_j^\perp}{\theta_j^\top - \theta_j^\perp} \geq b \\ &\iff \theta_j^i - \theta_j^\perp \geq b(\theta_j^\top - \theta_j^\perp) \\ &\iff \theta_j^i \geq b(\theta_j^\top - \theta_j^\perp) + \theta_j^\perp \\ &\iff \theta_j^i - b\theta_j^\top - (1-b)\theta_j^\perp \geq 0. \end{aligned} \tag{3.25}$$

3.3.2 Global queries

Global queries ask a user to consider preferences over full outcomes. We consider three types of global queries: anchor bound, anchor comparison and global comparison queries.

In certain domains, such as renting an apartment or choosing a travel package, outcomes have an associated *price*, measured in monetary units. Price plays a critical role in product choice, and is typically distinguished as an attribute. We make the standard assumption of *quasilinear utility* in which, overloading u, the utility $u(\mathbf{x}, p)$ of an outcome $\mathbf{x}$ obtained at price p is $u(\mathbf{x}, p) = \alpha u(\mathbf{x}) - p$. Here $u(\mathbf{x})$ is the price-independent utility of $\mathbf{x}$ and α is a valuation factor that adjusts u for currency. For quasilinear utilities, we can elicit strength of preference directly in terms of "willingness to pay," rather than in terms of lotteries between global outcomes (Engel and Wellman, 2007). Our global queries work in both lottery and willingness-to-pay settings.

3.3.2.1 Anchor bound queries (ABQs)

Anchor bound queries (ABQs) involve global factor *anchors* $\mathbf{b}^{j,\top}$ and $\mathbf{b}^{j,\bot}$, i.e., the best and worst outcomes with all attributes outside the factor (not just the conditioning set) fixed at their reference levels. The user is asked to specify whether the anchor utility $\theta_j^\top = u(\mathbf{b}^{j,\top})$ is greater than the specified bound b (similarly for $\mathbf{b}^{j,\bot}$). While such global bound queries could involve arbitrary outcomes, by limiting the queries to anchor outcomes we can arguably make such queries easier to answer, since most attributes (i.e., attributes outside the given factor) are fixed at reference levels and the attributes inside the factor are fixed to either best or worst levels. Bound queries allow us to impose constraints on parameters $\theta_j^\top$ and $\theta_j^\bot$ that feature prominently in the *local* GAI parameterization $u(\mathbf{x}^r) = \sum_{j=1}^{M} \left[(\theta_j^\top - \theta_j^\bot) \sum_{k=1}^{N_j} \mathbf{C}_j[r,k]\ v_j(\mathbf{x}_j^k) \right]$ (Eq. 3.20).

If we use standard lottery semantics, we can employ the bound b as the probability that calibrates the utility of $\mathbf{b}^{j,\top}$ and the standard lottery involving the worst possible outcome $\mathbf{x}^\bot$ and the best possible outcome $\mathbf{x}^\top$. A sample ABQ would be: "Consider the basic outcome

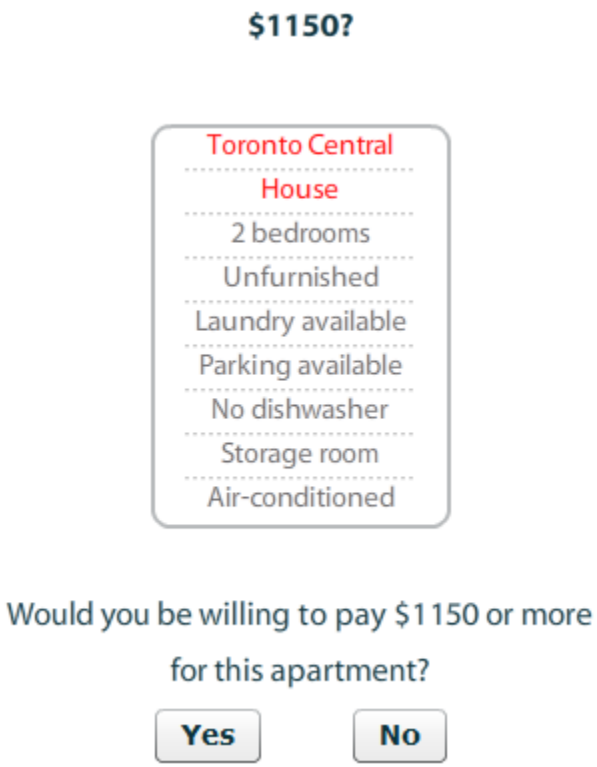


Figure 3.10: An anchor bound query used in the UTPREF recommendation system. The user is simply asked "Would you be willing to pay $1150 for the specified apartment?" The apartment is a top anchor for a factor with *Area* and *Housing type* attributes. All the attributes outside the factor are set to their reference values (identified by their grey font).

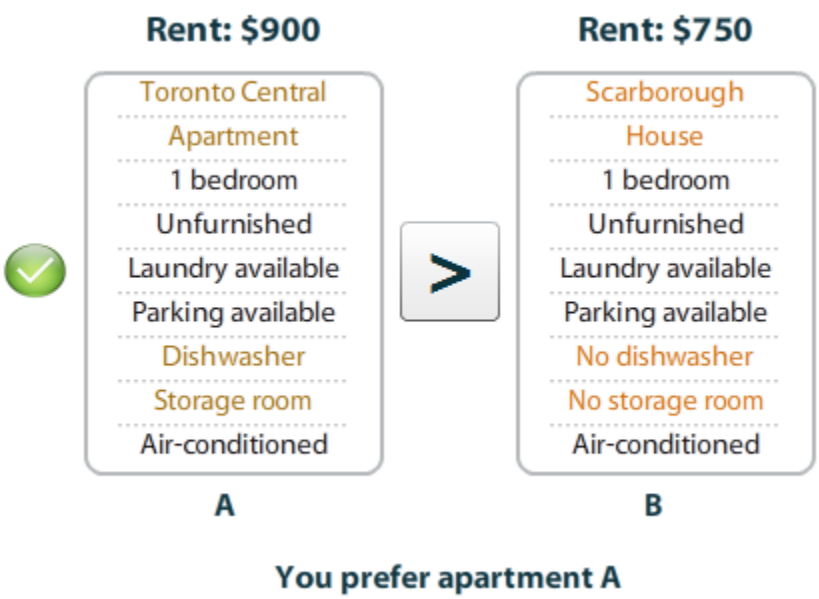


Figure 3.11: A global comparison query (with price) used in the UTPREF recommendation system. Given two outcomes, the user is simply asked to select the better option. Attributes that are the same for both outcomes are presented in black font, to aid the user in the comparison of the outcomes.

$\mathbf{b}^{j,\top}$, where attributes in factor F_j are set to their best values, and other attributes are fixed at reference levels. Would you prefer $\mathbf{b}^{j,\top}$ to a lottery $\langle b, \mathbf{x}^\top; 1-b, \mathbf{x}^\perp \rangle$?" A "yes" response gives $u(\mathbf{b}^{j,\top}) \geq b$, and therefore $\theta_j^\top \geq b$. If response is "no", $\theta_j^\top < b$. An analogous query exists for the "bottom" anchor $\mathbf{b}^{j,\perp}$.

Alternatively, in appropriate domains, a monetary scale can be used to calibrate global outcome utilities. Assuming $\alpha = 1$, in $u(\mathbf{x}, p) = \alpha u(\mathbf{x}) - p$, a sample query would be "Consider the basic outcome $\mathbf{b}^{j,\top}$, where attributes in factor F_j are set to their best values, and other attributes are fixed at reference levels. Would you be willing to pay \$b or more for this outcome?" As before, "yes" response would imply $\theta_j^\top \geq b$, and a "no" response $\theta_j^\top < b$. Fig. 3.10 shows a screenshot of an ABQ used in the UTPREF recommendation system.

3.3.2.2 Global comparison queries (GCQs and GCQPs)

Global comparison queries (GCQs) and *global comparison queries with price* (GCQPs) ask a user to compare two *arbitrary* outcomes $\mathbf{x}$ and $\mathbf{y}$, either taking into account their price at-

tributes $p_{\mathbf{x}}$ and $p_{\mathbf{y}}$ (GCQP) or ignoring them (GCQ). As long as the user can account for all the differences in attribute values (since for arbitrary outcomes, all attributes might be different), GCQs are simple and intuitive queries. They can be useful in applications where the number of attributes is not large. As we will discuss in Chapters 5 and 6, global comparison queries are very effective if the two outcomes are "current solution" outcomes, resulting from the minimax regret solution. An example GCPQ used in the UTPREF system is shown in Fig. 3.11.

As with all the queries discussed in this section, a response to a global comparison query imposes linear constraints on GAI parameters, potentially constraining many parameters at once:

$$\begin{aligned}(\mathbf{x}, p_{\mathbf{x}}) \succeq (\mathbf{y}, p_{\mathbf{y}}) &\iff u(\mathbf{x}, p_{\mathbf{x}}) \geq u(\mathbf{y}, p_{\mathbf{y}}) \\ &\iff \sum_{j=1}^{M} \sum_{k=1}^{N_j} C^k_{\mathbf{x}_j}\ \theta^k_j - p_{\mathbf{x}} \geq \sum_{j=1}^{M} \sum_{k=1}^{N_j} C^k_{\mathbf{y}_j}\ \theta^k_j - p_{\mathbf{y}} \\ &\iff \sum_{j=1}^{M} \sum_{k=1}^{N_j} (C^k_{\mathbf{x}_j} - C^k_{\mathbf{y}_j})\ \theta^k_j \geq p_{\mathbf{x}} - p_{\mathbf{y}}. \end{aligned} \tag{3.26}$$

Since $C^k_{\mathbf{x}_j}$, $C^k_{\mathbf{y}_j}$, $p_{\mathbf{x}}$ and $p_{\mathbf{y}}$ are known, the resulting inequality is a linear constraint on GAI parameters θ^k_j. For GCQ queries, where the price attribute is not considered, we can assume that $p_{\mathbf{x}} = p_{\mathbf{y}}$.

3.3.2.3 Anchor comparison queries (ACQs)

A particular form of GCQ is the *anchor comparison query* (ACQ), in which *both* outcomes to be compared are either top or bottom anchors for some (different) factors. ACQs are likely easier to understand because, unlike general GCQs, most attributes are fixed at reference levels, which are stable and salient. Responses to ACQs lead to linear constraints that involve only two GAI parameters (whereas responses to general GCQs or GCPQs tie together multiple parameters):

$$\mathbf{b}^{j,\top} \succeq \mathbf{b}^{k,\top} \iff \theta^{\top}_j \geq \theta^{\top}_k. \tag{3.27}$$

In the UTPREF system, anchor queries are visually similar to GCPQ queries (Fig. 3.11), with the exception of the price attribute.

3.4 Conclusion

3.4.1 Related work

Most previous work on utility elicitation traditionally assumed additive utility functions, which require strong preferential independence assumptions. Despite such limitations of additive models, elicitation (and even representation) of more flexible utility models, such as GAI, is a relatively new research area.

Decision-theoretic foundations

The GAI utility model was introduced in the 1960's by Fishburn (1967b, 1970). Although a known utility model within the decision analysis community (Keeney and Raiffa, 1976), it was rarely (to our knowledge) used in practical applications. Bacchus and Grove (1995) reintroduced the model to the AI community, by highlighting the similarities between compact representations of probability models, such as Bayesian networks, and GAI models in the utility domain. More recently, the analogy is further explored by Brafman and Engel (2009, 2010) by equating reference utilities with marginal probabilities.

Representation, optimization, and elicitation

In the last decade, GAI utility models have gained popularity within the AI community. Earlier work emphasizes user preference representation, rather than elicitation. Chajewska et al. (2000) introduce a real-life prenatal testing domain with five attributes, and discuss a possible GAI decomposition of preferences over the attributes. Hovewer, the Bayesian elicitation framework presented in the paper relies on a flat (not factored) utility representation, with direct utility queries over full outcomes. A related paper (Chajewska and Koller, 2000) uses the GAI

framework to estimate a probabilistic density function over utilities from a database of known utility functions (this approach will be discussed in more detail below); again, elicitation of individual utilities is not considered. More recently, there is renewed interest in *optimization* and aggregation of fully specified GAI preferences (Gonzales, Perny, and Queiroz, 2006, 2008; Dubus, Gonzales, and Perny, 2009a,b,c); however, optimization only becomes a bottleneck in relatively large domains with complex dependencies among GAI factors.

GAI representation of user preferences is a central model in the work of Boutilier et al. (2001) and Boutilier et al. (2003b) on minimax regret-based outcome optimization with incompletely specified utility functions. The subsequent extensions of this approach (Boutilier et al., 2001, 2005, 2006) focus on *elicitation* of GAI utilities, but without recognition of the decision-theoretic semantic issues of local queries.

Gonzales and Perny (2004) are one of the first to acknowledge the difficulties of GAI utility elicitation and provide a semantically sound elicitation procedure. Their approach, however, relies on using global queries with full (rather than local) outcomes, thus skirting the problem of local elicitation and global calibration. By resorting to full outcome queries, we lose some of the advantages of additive models and fail to exploit the decomposition of utility functions during the elicitation process (Braziunas and Boutilier, 2005).

Engel and Wellman (2007, 2010) a develop an iterative multiattribute auction protocol in which trader preferences are specified using GAI utility models. The traders' preferences are acquired through their bids on GAI factor instantiations.

Structure assessment

One assumption that we make in this thesis is that a user's utility function *structure* is known, or can be approximated well by a domain expert. Of course, such an assumption is not always realistic, since preferential dependencies between attributes can vary considerably from user to user. Ideally, an elicitation process would combine assessing preference structure (i.e., determining what sets of attributes are additively independent) with the acquisition of numerical

utility parameter values. The process would be optimized so that the user is asked only relevant questions about preferential independence between certain sets of attributes.

Solving such a problem is a difficult task. This is one reason that elicitation and verification of GAI model structure has not received adequate attention in the research literature (apart from early decision analysis texts by Fishburn (1970); Keeney and Raiffa (1976)). We mention the few attempts to address the issue.

Chajewska and Koller (2000) tackle the issue of automatic GAI structure discovery from data. They assume a database of full explicit utility functions (with possibly missing values), and present a framework for a Bayesian learning (using EM) of a Gaussian mixture model, where mixture components correspond to user subpopulations (types) with the same GAI utility structure. If the component structures are not fixed beforehand, the best-fitting structures are found by a greedy hill-climbing search algorithm over possible GAI factorizations. The search operators can add or subtract variables from factors, or introduce new factors with new variables.

A conceptually similar approach of using utility function values as data points, and then trying to find the best GAI structure that fits the data, is used by Engel and Wellman (2008b, 2010) to empirically evaluate the benefits of more flexible GAI models versus simple additive models. Both settings, however, are quite different from ours (although in some domains, the learning approach is complementary), where the goal would be to elicit the structure from individual users by posing structure queries about utility independencies between sets of attributes.

Baqui (2007) sets out to test whether GAI factors correspond to groupings of domain attributes according to their association with *fundamental objectives*. Fundamental objectives are higher-order additively-independent objectives that subsume domain attributes, according to the *value-focused thinking* paradigm of Keeney (1992). The task of specifying how domain attributes fulfill a user's fundamental objectives is relatively straightforward. The goal is to see whether the attribute factors induced by fundamental objectives satisfy the independence assumptions of a GAI model derived from the value tree. The results from a user study with

a real-estate domain are negative, showing that the concepts of fundamental objectives and generalized additive independence are orthogonal.

Finally, Brafman and Engel (2009, 2010) attempt to provide a principled way for structure elicitation by imposing directional constraints on a GAI graph, such that (by analogy to Bayesian networks) a node is *conditionally directionally independent* (CDI) from its ancestors given its parents.[1] The building of a directional network depends on the order of attributes, and the ability of a user to identify previous attributes (nodes) as parents of a current node. Thus, in the end, the structure elicitation burden is transfered to the user (by making it a structure specification problem), and some of the main issues in structure utility elicitation still remain the area for future research.

3.4.2 Contributions

The main contribution of this chapter is the presentation of a locally parameterized GAI model that, together with appropriate types of queries introduced in Section 3.3, can be used for effective and decision-theoretically sound elicitation of GAI utilities.

To design proper elicitation techniques for GAI models, we first have to solve the problem of subutility function semantics. Unlike additive models, GAI models require much more care in calibration because of the possible overlap of factors (sharing of attributes). If factors overlap, there are infinitely many valid decompositions of the same utility function in which the subutility functions vary considerably (i.e., not simply through some positive affine transformation). It is quite possible that the apparent "local preferences" for factor instantiations can be reversed in two different valid representations. Our identification of this problem (Braziunas and Boutilier, 2005) is the first contribution of the chapter, described in Section 3.2.1.

Our solution to this problem rests on using Fishburn's original canonical representation of subutility functions (Fishburn, 1967b) as a basis for decision-theoretically sound representation and elicitation of GAI utilities. In Section 3.2, we introduce a *parameterized* representation

[1] CDI is similar to utility independence, but is a weaker condition.

of GAI utilities (Eq. 3.15), analyze its properties, and provide a graphical search algorithm (Fig. 3.5) for computing the GAI structure parameters. In addition, our canonical GAI representation preserves the local structure of additive models. By taking into account the *conditioning sets* of attributes that shield the influence of other attribute values on local preferences over factor instantiations (analogously to a Markov blanket in a probabilistic graphical model), we generalize semantically sound local value functions for GAI models as well. As in additive models, the LVFs calibrate local preferences relative to the best and worst factor suboutcomes, assuming fixed values of the attributes in the conditioning set. The LVFs are local, because they involve only attributes in single factors and their (usually small) conditioning sets.

Using our representation, GAI models can be elicited by using both local queries about preferences over small subsets of attributes and global queries for calibration across utility factors. In Section 3.3, we present several types of semantically sound local and global queries that are both easy for users to understand and respond, and result in linear constraints on GAI utility parameters. Such queries are used in elicitation procedures described in the following chapters.

In the following Chapters 4 and 5 we consider elicitation strategies for selecting the right query during the elicitation process. Chapter 6 describes a user study that evaluated different types of GAI queries in terms of user-friendliness and cognitive cost in a realistic apartment search scenario.

Chapter 4

Bayesian elicitation

Contents

Representation of uncertainty over user preferences is a foundational issue in preference elicitation, affecting decision and query selection criteria as well as algorithmic methods for solving the problem. When uncertainty over utilities is quantified probabilistically, Bayesian principles can be applied to both decision making under partial utility information and elicitation of user preferences.

In this chapter, we describe Bayesian preference elicitation with GAI utility functions (Braziunas and Boutilier, 2005).

4.1 Decisions with partial utility information

In a Bayesian paradigm, utility functions are modeled as random variables drawn from a prior distribution; so, even though the decision support system does not know the user's exact preferences, it has *probabilistic* information about utility function parameters. The system's "beliefs" are updated using Bayes' rule, and the value of any decision or query is estimated by taking expectations with respect to possible user utility functions.

4.1.1 Decision criterion: maximum expected utility

In the general decision scenario that we consider in this thesis, a user (also referred to as a decision maker) can obtain an outcome from a finite set of possible outcomes (or alternatives) by choosing a decision from a finite set of available decisions. The effects of decisions are possibly stochastic; therefore, each decision can be viewed as a simple lottery over possible outcomes (see Section 2.1.2). The user has preferences over outcomes that can be represented

by a utility function. The optimal decision is the one that has the highest expected utility of the resulting outcomes (with the expectation taken over probabilities of decision outcomes).

In many decision scenarios with large and complex decision spaces (such as making travel plans or choosing an apartment to rent from hundreds of available options), selecting the optimal decision can require a lot of time and effort on the part of the user. A decision support system can help the decision maker by recommending a decision based on its knowledge of the decision scenario and the available knowledge of the user's preferences. While the set of outcomes and available decisions are often the same for many users, preferences over outcomes can vary substantially from user to user. To recommend good decisions for a particular decision maker, the decision support system has to gather enough relevant information about her preferences. In the following sections, we consider how to optimize the acquisition of preference information by posing a sequence of preference queries to the user. However, even when the decision support system has the ability to ask preference queries, obtaining the user's complete utility function is generally infeasible. Therefore, we first describe a model for recommending decisions with *partial* preference information.

Formally, the decision scenario consists of a finite set of possible outcomes $\mathbf{X}$, a finite set of decisions D, and a normalized utility function $u : \mathbf{X} \mapsto [0, 1]$. Decisions can have stochastic effects over outcomes. Each decision $d \in D$ results in an outcome $\mathbf{x} \in \mathbf{X}$ with probability $P_d(\mathbf{x})$. Thus, each decision d can be viewed a simple lottery (i.e., a discrete probability distribution) P_d over the outcome space $\mathbf{X}$ (see Section 2.1.2). By the expected utility theorem (Eq. 2.8), the utility of decision d with respect to the utility function u is the expected utility of its outcomes:

$$u(d) = \mathbb{E}_{\mathbf{x}\sim P_d}[u(\mathbf{x})] = \sum_{\mathbf{x}\in\mathbf{X}} P_d(\mathbf{x})u(\mathbf{x}). \tag{4.1}$$

In this chapter, we assume that the decision support system has prior *beliefs* about the user's utility function in a form of a probability distribution or density π over the space $\mathbf{U}$ of

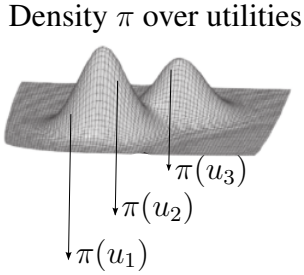


Figure 4.1: Expected utility of a decision given density π representing utility function uncertainty.

all possible utility functions. The value of decision d given density π over $\mathbf{U}$ is the expected utility of d.

Definition 4.1 (Expected utility of a decision given density over utilities) The expected utility of decision $d \in D$ given density π over all possible utility functions $\mathbf{U}$ is

$$EU(d,\pi) = \mathbb{E}_{u\sim\pi}[u(d)] = \int_{u\in\mathbf{U}} \pi(u)\, u(d) du. \tag{4.2}$$

Figure 4.1 shows a visual example of the probabilistic representation of uncertainty over user utility functions.

The EU expression above is sensitive to positive affine transformations of utility functions. An important precondition for representing uncertainty in terms of probability distributions over a set of utility functions is for these functions to be *extremum equivalent*, i.e., share the same best and worst outcomes (Boutilier, 2003). Extremum equivalence is sufficient to put all utility functions on a common scale. In practice, it is quite reasonable to assume that the decision maker has the same best and worst outcomes under any utility function. The decision support system has to determine those outcomes before engaging in recommendation or elicitation of further preferences.

With a probability distribution over utilities, the decision support system can use a principled Bayesian criterion for optimal recommendations under any probability distribution π: the optimal decision d^* is the one that has the highest *expected* utility (with expectation over

possible utility functions u):[1]

$$d^* = \arg\max_{d \in D} EU(d, \pi) = \arg\max_{d \in D} \mathbb{E}_{u \sim \pi}[u(d)]. \tag{4.3}$$

The density π can be considered as the *belief state* that the decision support system is in with respect to its probabilistic knowledge about the user's utility function. We can define the value of being in a belief state π as the expected utility of the best decision the system can recommend given density π.

Definition 4.2 (Maximum expected utility of a belief state) The value of *belief state* π is the expected utility of the best decision given π:

$$MEU(\pi) = EU(d^*, \pi) = \max_{d \in D} EU(d, \pi). \tag{4.4}$$

4.1.2 GAI models

4.1.2.1 Decision scenario

In the remaining chapters of this thesis (including the current one), we focus on decision scenarios in which

1) the outcome space $\mathbf{X}$ is multiattribute, that is, $\mathbf{X} = X_1 \times \cdots \times X_N$ is the set of all instantiations of N attributes $X_1, X_2, \ldots, X_N$ with finite domains;

2) all decisions are deterministic: $D \subseteq \mathbf{X}$;

3) user preferences are generalized additively independent, and, therefore, can be represented by a GAI utility function.

In practice, outcomes are often endowed with multiattribute structure (or can be modeled as such). GAI utilities provide a flexible representation framework for structured preferences over multiattribute outcomes; they are less restrictive and, therefore, more widely applicable

[1] Considering that the utility of a decision is the expected utility of its outcomes, Boutilier (2002, 2003) calls this criterion the *expected expected utility* criterion, with double expectation over utility functions and outcomes.

than additive utilities. All the results in this chapter can be extended to settings where decisions have stochastic effects. We limit the set of decisions to deterministic choices to simplify the exposition of the material in this chapter, and to maintain consistency with the decision settings in Chapters 5 and 6, where decisions are viewed as feasible configurations or selections from a database of multiattribute items.

With deterministic decisions, each decision $d \in D$ corresponds to some *feasible* outcome $\mathbf{x} \in \mathbf{X}_F$, where $\mathbf{X}_F \subseteq \mathbf{X}$ is the set of all feasible (obtainable) outcomes. Following Eq. 4.2, the value of outcome $\mathbf{x}$ given density π over utilities is

$$EU(\mathbf{x}, \pi) = \mathbb{E}_{u\sim\pi}[u(\mathbf{x})] = \int_{u\in\mathbf{U}} \pi(u)\, u(\mathbf{x}) du. \tag{4.5}$$

The optimal outcome $\mathbf{x}^*$ has the highest *expected* utility (Eq. 4.3):

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathbf{X}_F} EU(\mathbf{x}, \pi) = \arg\max_{\mathbf{x}\in\mathbf{X}_F} \mathbb{E}_{u\sim\pi}[u(\mathbf{x})]. \tag{4.6}$$

With deterministic actions, the value of being in belief state π is the expected utility of the best outcome given π:

$$MEU(\pi) = EU(\mathbf{x}^*, \pi) = \max_{\mathbf{x}\in\mathbf{X}_F} EU(\mathbf{x}, \pi). \tag{4.7}$$

4.1.2.2 Notation

We provide a brief recap of notation introduced in Chapter 3. As before, we have the outcome set $\mathbf{X} = X_1 \times \cdots \times X_N$ defined by instantiations of N attributes $X_1, X_2, \ldots, X_N$, each with finite domains. Given an index set $I \subseteq \{1, \ldots, N\}$, we define $\mathbf{X}_I = \times_{i\in I} X_i$ to be the set of *partial outcomes* (or *suboutcomes*) restricted to attributes indexed by I. We also assume a collection of M attribute subsets, or factors, that cover the set of all attributes: $F_1 \cup F_2 \cdots \cup F_M = \{X_1, X_2, \ldots, X_N\}$. A factor $F_I = \{X_i \mid i \in I\}$ contains the attributes whose indices are in the index set $I \subseteq \{1, \ldots, N\}$. For a factor F_j, $\mathbf{x}_{I_j}$, or simply $\mathbf{x}_j$, is a particular instantiation of its attributes. The factors (and their associated sets of indices) are enumerated from 1 to M: $F_1, F_2, \ldots, F_M$. To simplify the notation, we write $F_j = F_{I_j}$ (and $\mathbf{x}_j = \mathbf{x}_{I_j}$).

Let $\mathbf{x}^0 = (x_1^0, x_2^0, \ldots, x_N^0)$ denote a special outcome in $\mathbf{X}$ designated as the *reference outcome* (see Section 3.1.3). By using a reference outcome, any outcome $\mathbf{x} \in \mathbf{X}$ can be "projected" to a subset space indexed by I, resulting in the outcome $\mathbf{x}[I]$. For any $\mathbf{x} \in \mathbf{X}$, $\mathbf{x}[I]$ is an outcome where attributes of $\mathbf{x}$ not indexed by I are clamped at the reference values.

Each factor has $N_j = |\mathbf{X}_j|$ local (partial) outcomes, i.e., possible instantiations of factor attributes. A *basic outcome* $\mathbf{b}$ for factor F_j is any outcome $\mathbf{x}$ with attributes outside factor F_j set to the reference level: $\mathbf{b} = \mathbf{x}[I_j]$ for some $\mathbf{x} \in \mathbf{X}$. Each factor F_j has N_j basic outcomes, corresponding to N_j partial outcomes of F_j. The k^{th} basic outcome for factor F_j is denoted as $\mathbf{b}^{j,k}$.

The GAI factors are assumed to be generalized additively independent (see Definition 3.1); therefore, user preferences can be represented by a GAI utility function $u(\mathbf{x}) = \sum_{j=1}^{M} u_j(\mathbf{x}_j)$ (Theorem 3.1).

4.1.2.3 Two parametric representations of GAI utilities

In Chapter 3, we introduced two parameterized GAI model representations that rely on the notions of GAI parameters, local value parameters, and structure coefficients. *GAI parameters* (also referred to as *basic outcome parameters*) $\theta_j^k = u(\mathbf{b}^{j,k})$ represent utility values of basic outcomes. We let $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ to denote a vector of all basic outcome parameter values ($\boldsymbol{\Theta}$ is the space of all possible GAI parameter vectors):

$$\boldsymbol{\theta} = (\theta_1^1, \theta_1^2, \ldots, \theta_j^1, \theta_j^2, \ldots, \theta_j^{N_j}, \ldots, \theta_M^{N_M}).$$

Structure coefficients encode attribute decomposition into factors (see Section 3.2.2 for details). For each factor F_j, $\mathbf{C}_j$ is a square $N_j \times N_j$ matrix of integer coefficients, whose rows specify the linear combination of GAI parameters defining factor subutility values. For each factor outcome $\mathbf{x}_j$, $\mathbf{C}_{\mathbf{x}_j}$ is the $ind(\mathbf{x}_j)$ row of $\mathbf{C}_j$ ($ind(\mathbf{x}_j)$ is the index of the local configuration $\mathbf{x}_j$), and $C_{\mathbf{x}_j}^k = C_j[ind(\mathbf{x}_j), k]$ is the k^{th} entry in that row vector.

The first parameterization (Eqs. 3.15, 3.22) defines the utility of any outcome $\mathbf{x} \in \mathbf{X}$ as a

linear combination of GAI parameters:

$$u(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^{M} \sum_{k=1}^{N_j} C^k_{\mathbf{x}_j}\ \theta^k_j. \tag{4.8}$$

This representation is *global*, in the sense that all the parameters are semantically defined with respect to full outcomes (GAI parameters are utilities of basic outcomes, and basic outcomes are full outcomes).

The second GAI representation uses mostly local LVF parameters. An *LVF (local value function) parameter* $v^k_j = v_j(\mathbf{x}^k_j)$ denotes the local value of factor configuration $\mathbf{x}^k_j$. Such local value parameters can be assessed by local queries that consider only attributes in the relevant factor and its conditioning set (see Section 3.3). The local GAI representation also uses a small number ($2M$) of global *anchor parameters* $\theta^\top_1, \theta^\perp_1, \ldots, \theta^\top_M, \theta^\perp_M$ that are necessary for utility calibration across different factors. The global parameters $\theta^\top_j = u(\mathbf{b}^{j,\top})$ and $\theta^\perp_j = u(\mathbf{b}^{j,\perp})$ define the utilities of the best and worst basic outcomes for factor F_j. The local parameterization of the GAI utility function (Eq. 3.20, 3.23) is as follows:

$$u(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^{M} \left[(\theta^\top_j - \theta^\perp_j) \sum_{k=1}^{N_j} C^k_{\mathbf{x}_j}\ v^k_j \right]. \tag{4.9}$$

Since structure coefficients $\mathbf{C}_j$ are fixed given the GAI factor decomposition, a GAI utility function is fully determined by either the global parameters $\{\theta^1_1, \ldots, \theta^k_j, \ldots, \theta^{N_M}_M\}$, or by the $2M$ global parameters $\{\theta^\top_1, \theta^\perp_1, \ldots, \theta^\top_M, \theta^\perp_M\}$ and the local LVF parameters $\{v^1_1, \ldots, v^k_j, \ldots, v^{N_M}_M\}$. Global GAI parameters and local LVF parameters are linearly related *if anchor parameters are fully known*:

$$\begin{aligned} \theta^k_j &= (\theta^\top_j - \theta^\perp_j)\, v^k_j + \theta^\perp_j, \\ v^k_j &= \frac{1}{\theta^\top_j - \theta^\perp_j}\, \theta^k_j - \frac{\theta^\perp_j}{\theta^\top_j - \theta^\perp_j}. \end{aligned} \tag{4.10}$$

Therefore, information about local parameters can be used to assess global parameters, and vice versa.

4.1.2.4 Uncertainty representation

With GAI models, we assume that uncertainty over utility functions is specified by probability densities over GAI utility function parameters. Let π_θ be a density over the space of GAI parameters $\boldsymbol{\Theta}$. Then, using Eq. 4.8 and linearity of expectation, the expected utility of outcome $\mathbf{x} \in \mathbf{X}$ becomes:

$$EU(\mathbf{x}, \pi_\theta) = \mathbb{E}_{\boldsymbol{\theta}}[u(\mathbf{x}|\boldsymbol{\theta})] = \mathbb{E}[\sum_{j=1}^{M} \sum_{k=1}^{N_j} C^k_{\mathbf{x}_j}\, \theta^k_j] = \sum_{j=1}^{M} \sum_{k=1}^{N_j} C^k_{\mathbf{x}_j}\, \mathbb{E}[\theta^k_j]. \tag{4.11}$$

Using the second GAI parameterization (Eq. 4.9), and assuming a prior distribution π_v over the space of LVF parameters v^k_j, the expected utility of outcome $\mathbf{x}$ can be written as

$$EU(\mathbf{x}, \pi_v) = \mathbb{E}[\sum_{j=1}^{M} (u^\top_j - u^\perp_j)\, \sum_{k=1}^{N_j} C^k_{\mathbf{x}_j}\, v^k_j\,] = \sum_{j=1}^{M} (u^\top_j - u^\perp_j)\, \sum_{k=1}^{N_j} C^k_{\mathbf{x}_j}\, \mathbb{E}[v^k_j]. \tag{4.12}$$

The two settings are equivalent if anchor parameters are fully known. In such a case (see Eq. 4.10),

$$\mathbb{E}[\theta^k_j] = (\theta^\top_j - \theta^\perp_j)\, \mathbb{E}[v^k_j] + \theta^\perp_j,$$
$$\mathbb{E}[v^k_j] = \frac{1}{\theta^\top_j - \theta^\perp_j}\, \mathbb{E}[\theta^k_j] - \frac{\theta^\perp_j}{\theta^\top_j - \theta^\perp_j}.$$

Situations where the local values are uncertain, but anchor utilities are fully known, have been commonly assumed with additive utilities (Fishburn, 1964; Sarin, 1977; Kirkwood and Sarin, 1985; Hazen, 1986; Weber, 1987; White et al., 1983, 1984; Anandalingam and White, 1993; Blythe, 2002). While somewhat restrictive, the setting allows us to use the local parameterization (Eq. 4.9) that is amenable to elicitation using local queries. Global anchor parameters have to be fully assessed beforehand. This can be done by using global standard gamble queries that involve the best and the worst outcomes $\mathbf{x}^\top$, $\mathbf{x}^\perp$ (e.g., "What is the probability p for which you would be indifferent between outcome $\mathbf{b}^{j,\top}$ and the lottery $\langle p, \mathbf{x}^\top;\, 1-p, \mathbf{x}^\perp\rangle$?") or, in willingness-to-pay domains, using queries that calibrate against a monetary scale (e.g., "How much would you be willing to pay for outcome $\mathbf{b}^{j,\top}$?"). A combination of global queries, including direct, bound and comparison queries, can also be used to assess anchor parameters (see Section 3.3).

4.1.2.5 Optimization in GAI models

In the remainder of this chapter, we will assume that anchor parameters are known, and uncertainty over utilities is specified by the density π_v over the space of LVF parameters v_j^k. As shown above in Eq. 4.12, the expected utility of outcome $\mathbf{x}$ is then

$$EU(\mathbf{x}, \pi_v) = \sum_{j=1}^{M} (u_j^{\top} - u_j^{\perp}) \sum_{k=1}^{N_j} C_{\mathbf{x}_j}^k \, \mathbb{E}[v_j^k].$$

The optimal outcome $\mathbf{x}^*$ is the one with the highest expected utility. If the means $\mathbb{E}[v_j^k]$ of LVF parameters are known, the optimal outcome $\mathbf{x}^*$ can be found in the same way as with fully specified GAI utility functions. Let $u_j^E(\mathbf{x}_j) = (u_j^{\top} - u_j^{\perp}) \sum_{k=1}^{N_j} C_{\mathbf{x}_j}^k \, \mathbb{E}[v_j^k]$. Then,

$$EU(\mathbf{x}, \pi_v) = \sum_{j=1}^{M} (u_j^{\top} - u_j^{\perp}) \sum_{k=1}^{N_j} C_{\mathbf{x}_j}^k \, \mathbb{E}[v_j^k] = \sum_{j=1}^{M} u_j^E(\mathbf{x}_j).$$

Thus, finding the optimal outcome $\mathbf{x}^*$ with uncertainty π_v over LVF parameters is equivalent to finding the optimal outcome in a fully specified GAI function u^E with subutilities $u_j^E(\mathbf{x}_j) = (u_j^{\top} - u_j^{\perp}) \sum_{k=1}^{N_j} C_{\mathbf{x}_j}^k \, \mathbb{E}[v_j^k]$:

$$\mathbf{x}^* = \operatorname*{argmax}_{\mathbf{x} \in \mathbf{X}_F} EU(\mathbf{x}, \pi) = \operatorname*{argmax}_{\mathbf{x} \in \mathbf{X}_F} u^E(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{x} \in \mathbf{X}_F} \sum_{j=1}^{M} u_j^E(\mathbf{x}_j). \tag{4.13}$$

This optimization problem can be solved efficiently by using *variable elimination* (Dechter, 1996), a form of non-serial dynamic programming (Bertele and Brioschi, 1972). Consider the following maximization problem:

$$\max_{\mathbf{x} \in \mathbf{X}_F} u(\mathbf{x}) = \max_{\mathbf{x} \in \mathbf{X}_F} \sum_{j=1}^{M} u_j(\mathbf{x}_j) = \max_{x_1} \max_{x_2} \ldots \max_{x_N} \sum_{j=1}^{M} u_j(\mathbf{x}_j). \tag{4.14}$$

By distributing the max operator inward over summations (using the distributive property of the max operator) and then "eliminating" one variable at a time, the variable elimination algorithm can avoid the exponential (in the number of variables) run time of the naive maximization procedure. Variable elimination efficiency depends on the variable elimination order; its run time is exponential in the *tree width* of the *join tree* induced by the variable elimination algorithm (Lauritzen and Spiegelhalter, 1988). In many GAI decompositions the tree width is

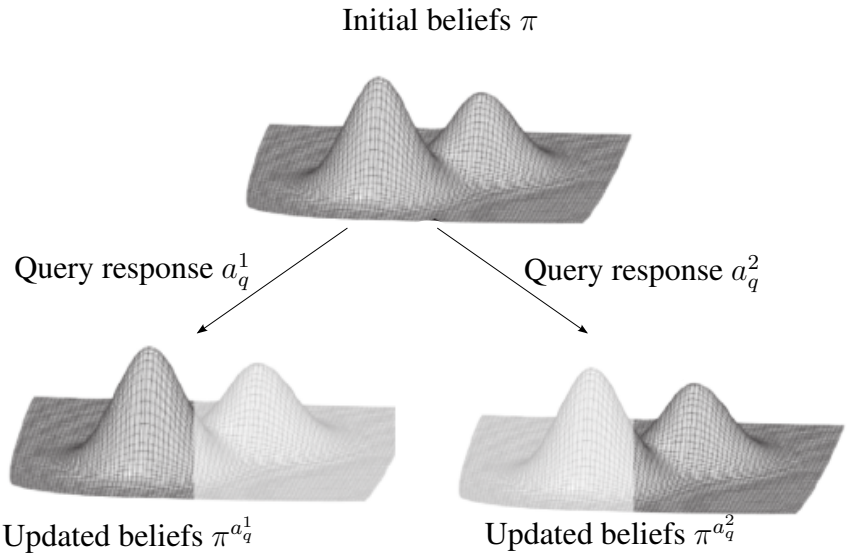


Figure 4.2: Update of beliefs π given two possible responses to query q.

much smaller than the total number of attributes. Therefore, finding the optimal outcome in GAI utility functions is a locally exponential procedure; if the tree width is bounded by a constant, the procedure is of (pseudo)polynomial time complexity (Braziunas and Boutilier, 2005; Boutilier et al., 2006).

4.2 Elicitation

In this section, we start by outlining the general elicitation framework that supports interaction between the decision support system and the decision maker. We then describe a natural Bayesian criterion, based on the notion of expected value of information, for selecting myopically optimal queries. We extend the elicitation framework to GAI utilities by assuming that uncertainty over utilities is represented by a density over LVF parameters; we limit the set of available queries to local bound queries. Finally, we design a tractable algorithm for myopically optimal elicitation by adopting a mixture-of-uniforms density representation.

4.2.1 Elicitation framework

With probabilistic knowledge of a decision maker's utilities, a decision support system can evaluate the expected utility of each available option and recommend the outcome with the highest expected utility. When the system's knowledge of the user's preferences is limited, the recommended outcome might be of much lower value than the optimal outcome (with respect to the user's true utility function). By obtaining additional preference information, the system can improve its recommendations. We model this scenario by allowing the decision support system to ask a series of questions about the user's preferences; the system can use the user's responses to reduce its uncertainty about the user's utility function. Formally, we assume a finite set of available *queries* Q, and, for each query $q \in Q$ — a set of possible user *responses* (answers) A_q. Responses to queries depend on the true user utility function u, but might be noisy. This relationship is captured by the probabilistic *response model* $Pr(a_q|u)$ that specifies the probability of response a_q to query q when utility function is u.

As before, we assume that the system's beliefs about user utilities are summarized by the probability density π over $\mathbf{U}$. When the true user utility function is not fully known, the decision support system estimates the probability of a particular response to the given query by considering the response's likelihood under every possible utility function:

$$\Pr(a_q|\pi) = \mathbb{E}_{u\sim\pi}[Pr(a_q|u)] = \int_{u\in\mathbf{U}} \Pr(a_q|u)\pi(u)du. \tag{4.15}$$

A response a_q to query q provides information about the true utility function and changes the current belief state from π to π^{a_q} according to the Bayes' rule:

$$\pi^{a_q}(u) = \pi(u|a_q) = \frac{Pr(a_q|u)\pi(u)}{\int_{u\in\mathbf{U}} \Pr(a_q|u)\pi(u)du}. \tag{4.16}$$

Figure 4.2 illustrates the update of utility function beliefs given two possible responses to a preference query.

Elicitation of preferences takes time, imposes cognitive burden on users, and might involve considerable computational and financial expense. Such factors are modeled by assigning each

Input: Set of feasible outcomes $\mathbf{X}_F$, set of queries Q, set of query costs $\{c_q|q \in Q\}$, sets of responses $\{A_q|q \in Q\}$, response model $Pr(a_q|u)$, prior beliefs π over user utilities $\mathbf{U}$, termination criteria T
Output: Recommended outcome $\mathbf{x}^*$

$\mathbf{x}^* \leftarrow \text{argmax}_{\mathbf{x}\in\mathbf{X}_F} EU(\mathbf{x}, \pi)$
while *termination criteria T not met* **do**
  select query q
  pose query q to the user
  receive response a_q
  update the system's beliefs to π^{a_q} according to Eq. 4.16
  $\pi \leftarrow \pi^{a_q}$
  $\mathbf{x}^* \leftarrow \text{argmax}_{\mathbf{x}\in\mathbf{X}_F} EU(\mathbf{x}, \pi)$
end

Figure 4.3: A generic procedure that supports Bayesian preference elicitation through a sequence of interactions (queries and responses) between the decision support system and the user (decision maker).

query q a *query cost* c_q.[1]

We can now outline the generic preference elicitation procedure. In its basic form, the decision support system repeatedly selects a preference query from the set Q, receives a user response, updates its current beliefs about the user's utility function, and continues until termination criteria are met. Termination criteria depend on a specific decision scenario. Simple termination criteria include exceeding the maximum number of queries allowed, or the cumulative cost of queries reaching some predefined threshold. More complex termination criteria utilize the notion of expected utility loss, or regret, which is the expected (with respect to π) utility difference between the optimal outcome and the recommended outcome. Let $\mathbf{x}^*_\pi = \max_{\mathbf{x}\in\mathbf{X}_F} \mathbb{E}_{u\sim\pi}[u(\mathbf{x})]$ be the optimal outcome for belief state π. Then, the expected loss of recommending $\mathbf{x}^*_\pi$ is

$$\mathbb{E}_{u\sim\pi}[\max_{\mathbf{x}\in\mathbf{X}_F} u(\mathbf{x}) - u(\mathbf{x}^*_\pi)]. \tag{4.17}$$

Expected loss is usually hard to compute exactly. However, it can be approximated by sampling

[1] More generally, costs could depend on the true utility function, or could be associated with responses.

(Chajewska et al., 2000); this is what we do in the experiments in Section 4.3. The elicitation process stops when expected utility loss falls below a certain threshold (after accounting for query costs).

Figure 4.3 shows the generic Bayesian elicitation framework that is assumed in the remainder of this chapter.

4.2.2 Myopic elicitation

Our elicitation model is a sequential process whose value is determined by the expected utility of a terminal decision and information gathering costs. Ideally, a sequential elicitation policy takes into consideration all possible future questions and answers, and provides an optimal tradeoff between query costs (the burden of elicitation) and the potentially better decisions the system can recommend with additional preference information. As shown by Boutilier (2002), computing such a policy amounts to solving a partially observable Markov decision process (POMDP) with a continuous state space (described in Section 2.3.6.1).

Solving any POMDP is generally computationally difficult (even for flat utility models). Therefore, we focus on suboptimal, but tractable *myopic* elicitation strategies. Myopic elicitation is driven by queries that provide highest *expected value of information* (EVOI), where expectation is over possible realizations of user utility functions. By focusing on maximizing immediate EVOI, a myopic strategy ignores the potential value of future queries. Nonetheless, greedy strategies have been successfully used in real and synthetic domains (Chajewska et al., 2000; Braziunas and Boutilier, 2005). We should also note that by increasing the "lookahead" horizon, a myopic policy generally approaches the performance of a sequentially optimal policy.

Expected value of information of a preference query

The maximum expected utility of being in belief state π provides a myopic estimate of its value, because it only considers the expected utility of the best *immediate* decision the system

can recommend (Eq. 4.7):

$$MEU(\pi) = EU(\mathbf{x}^*, \pi) = \max_{\mathbf{x} \in \mathbf{X}_F} EU(\mathbf{x}, \pi).$$

A response a_q to query q provides information about the true utility function and changes the decision support system's belief state π to the new belief state π^{a_q} according to the Bayes' rule (Eq. 4.16). After response a_q, the new belief state's value is $MEU(\pi^{a_q})$.

To calculate the value of some query $q \in Q$, we can weigh the MEUs resulting from its possible responses according to their likelihood (given by the response model $Pr(a_q|u)$). The resulting measure of query value is called the *expected posterior utility* (EPU) of the query.

Definition 4.3 (Expected posterior utility (EPU))

Given query q, response model $Pr(a_q|u)$, and prior beliefs over utilities π, the expected posterior utility of q is

$$EPU(q, \pi) = \sum_{a_q \in A_q} \Pr(a_q|\pi)\ MEU(\pi^{a_q}), \tag{4.18}$$

where

$$\Pr(a_q|\pi) = \mathbb{E}_{u \sim \pi}[Pr(a_q|u)] = \int_{u \in \mathbf{U}} \Pr(a_q|u)\pi(u)du \qquad \text{(Eq. 4.15)},$$

$$\pi^{a_q}(u) = \pi(u|a_q) = \frac{Pr(a_q|u)\pi(u)}{\int_{u \in \mathbf{U}} \Pr(a_q|u)\pi(u)du} \qquad \text{(Eq. 4.16)}.$$

$EPU(q, \pi)$ is the expected utility of asking query q in belief state π. The gain in expected utility due to asking the query q (from $EU(\pi)$ to $EPU(q, \pi)$) is called the *expected value of information* (EVOI) of query q.

Definition 4.4 (Expected value of information (EVOI))

Given query q, and beliefs over utilities π, the *expected value of information* of query $q \in Q$ is

$$EVOI(q, \pi) = EPU(q, \pi) - MEU(\pi). \tag{4.19}$$

By selecting queries based on their EVOI (or, equivalently, EPU), the decision support system can implement a principled (although myopic) Bayesian elicitation procedure in which, at

Input: Set of feasible outcomes $\mathbf{X}_F$, set of queries Q, set of query costs $\{c_q|q \in Q\}$, sets of responses $\{A_q|q \in Q\}$, response model $Pr(a_q|u)$, prior beliefs π over user utilities $\mathbf{U}$, termination criteria T
Output: Recommended outcome $\mathbf{x}^*$

$\mathbf{x}^* \leftarrow \text{argmax}_{\mathbf{x}\in\mathbf{X}_F} EU(\mathbf{x}, \pi)$
while *termination criteria* T *not met* **do**
  $q^* \leftarrow \text{argmax}_{q\in Q} EVOI(q) - c_q$ (Eq. 4.19)
  pose query q^* to the user
  receive response a_{q^*}
  update the system's beliefs to $\pi^{a_{q^*}}$ according to Eq. 4.16
  $\pi \leftarrow \pi^{a_{q^*}}$
  $\mathbf{x}^* \leftarrow \text{argmax}_{\mathbf{x}\in\mathbf{X}_F} EU(\mathbf{x}, \pi)$
end

Figure 4.4: A myopic Bayesian preference elicitation procedure that uses a sequence of interactions (queries and responses) between the decision support system and the user (decision maker). The decision support system selects queries based on their expected value of information.

each step, the query with the highest value of EVOI minus query cost is asked, and the distribution over utilities is updated based on user responses; the process stops when the expected value of a decision meets some termination criteria. Figure 4.4 provides further details about the myopic Bayesian elicitation framework.

4.2.3 GAI models

In this section, we apply the myopic Bayesian elicitation framework to elicit GAI utility model parameters. As discussed in Section 4.1.2.4 above, we assume that uncertainty over a user's utility function is specified by the probabilistic prior π_v over GAI local value functions (LVFs); the factor anchors $\theta_1^\top, \theta_1^\perp, \ldots, \theta_M^\top, \theta_M^\perp$ are known. Let $\lambda_j = \theta_j^\top - \theta_j^\perp$ be the *scaling coefficient* for factor F_j. We also assume that all expectations are with respect to the distribution π_v: $\mathbb{E} = \mathbb{E}_{u\sim\pi_v}$. As shown in Eq. 4.12, the expected utility of outcome (or configuration) $\mathbf{x}$ is then

$$EU(\mathbf{x}, \pi_v) = \mathbb{E}[u(\mathbf{x})] = \sum_{j=1}^{M} \mathbb{E}[u_j(\mathbf{x}_j)] = \sum_{j=1}^{M} \lambda_j \sum_{k=1}^{N_j} C_{\mathbf{x}_j}^k \mathbb{E}[v_j^k].$$

To gain more information about the user's LVFs, we can ask appropriate *local* queries (see Section 3.3). Although local *comparison* queries are natural and simple to answer, they are

difficult to deal with in a Bayesian elicitation framework because there are no parametric probability distributions that remain conjugate to the prior after a response to a comparison query. Responses to comparison queries impose "diagonal" constraints on the prior distribution. For example, if $\pi(v_j^1, v_j^2)$ were a density over two LVF parameters v_j^1, v_j^2, then, after the response yes to the comparison query $q =$ "Do you prefer local outcome $\mathbf{x}_j^1$ to $\mathbf{x}_j^2$?", the posterior distribution would be $\pi^{yes_q} = \pi(v_j^1, v_j^2 | v_j^1 \geq v_j^2)$. To maintain the parametric form, π^{yes_q} would have to be approximated by, for example, sampling from the true posterior $\pi(v_j^1, v_j^2 | v_j^1 \geq v_j^2)$ and refitting to the samples using expectation maximization (EM) or similar techniques. *Bound* queries, on the other hand, impose only axis-parallel constraints on a single variable. They fit well with certain priors, such as mixtures of truncated Gaussians or uniforms, which remain closed under updates due to bound queries. For this reason, in this chapter we limit our query set to *local bound queries*.

Local bound queries, introduced in Section 3.3.1.3, ask the user to consider a single local outcome $\mathbf{x}_j^r$ (i.e., the r^{th} local configuration in factor F_j), and decide whether its LVF value $v_j^r = v_j(\mathbf{x}_j^r)$ is greater or less than some specified bound b. One way of asking LBQs relies on the standard lottery semantics to calibrate the value of $\mathbf{x}_j^r$ with respect to $\mathbf{x}_j^\perp$ and $\mathbf{x}_j^\top$ (the best and worst outcomes of factor F_j): "Assume that the attributes in the conditioning set of factor F_j are fixed at reference levels. Would you prefer the local outcome $\mathbf{x}_j^r$ to the lottery $\langle b, \mathbf{x}_j^\top; 1-b, \mathbf{x}_j^\perp \rangle$, ceteris paribus?" If the answer is "yes", $v_j(\mathbf{x}_j^r) \geq b$; if "no", then $v_j(\mathbf{x}_j^r) < b$. A practical approximation of the probabilistic semantics can be achieved by asking the user to simply consider the outcome $\mathbf{x}_j^r$ on a scale from 0 to 1 (or 0 to 100), where 0 corresponds to the worst local outcome $\mathbf{x}_j^\perp$ and 1 corresponds to the best outcome $\mathbf{x}_j^\top$, and specify if the value of $\mathbf{x}_j^r$ is greater or less than the bound b (of course, as in other local queries, the attributes in the conditioning set are assumed fixed at reference values). Such non-probabilistic LBQs are arguably easier to explain and possibly easier to answer. Fig. 3.9 shows an example of an LBQ used in the UTPREF recommendation system (described in Chapter 6).

An LBQ $q[v_j^r, b]$ that elicits the relationship between the local value v_j^r and the bound $b \in$

Figure 4.5: A local bound query used in the UTPREF recommendation system. The scale is 0 to 100 (corresponding to the LVF range of 0 to 1). Here, the user has indicated that the value of the `Toronto East, House` outcome is somewhere between 0 and 50 by dragging the outcome in question to the lower bin. The conditioning set attribute *Number of bedrooms* is set to its reference value `2 bedrooms` and is distinguished by the grey font color.

$[0, 1]$ is defined by three parameters: the factor index j, the local configuration index r, and the bound b. Its response set $A_q = \{yes_q, no_q\}$ consists of two responses yes and no. We use query subscripts to associate responses to queries (so yes_q is response yes to query q). In the remainder of this chapter, we assume that the set of available queries is limited to LBQs: $Q = \{q[v_j^r, b], j = 1..M, r = 1..N_j, b \in [0, 1]\}$.

4.2.3.1 Posterior distributions

For myopically optimal elicitation, we need to find which query has the highest expected posterior utility, defined as the expected (with respect to the possible responses) MEU of its updated belief states. The two belief states that result from yes and no responses to the LBQ $q[v_j^r, b]$ are π^{yes_q} and π^{no_q}. In this subsection, we show how to compute $MEU(\pi^{yes_q})$ and $MEU(\pi^{no_q})$ in GAI models *under the assumption that the random variables representing local value parameters v_j^r are independent*. This is a reasonable assumption for situations where the prior satisfies the assumption, and elicitation process employs only LBQs (which maintain independent posteriors). In the following subsection, we use the MEU expression to compute an LBQ's expected posterior utility.

For simplicity, we assume a noiseless user response model $Pr(a_q|u)$:

$$\Pr(yes_{q[v_j^r,b]}|u) = \begin{cases} 1, \text{ if } v_j^r \geq b, \\ 0, \text{otherwise.} \end{cases}$$

$$\Pr(no_{q[v_j^r,b]}|u) = 1 - \Pr(yes_{q[v_j^r,b]}|u).$$

The two posteriors π^{yes_q} and π^{no_q} are (Eq. 4.16):

$$\pi^{yes_q}(u) = \pi(u|yes_q) = \frac{\Pr(yes_q|u)\pi(u)}{\int_{u\in\mathbf{U}} \Pr(yes_q|u)\pi(u)du},$$
$$\pi^{no_q}(u) = \pi(u|no_q) = \frac{\Pr(no_q|u)\pi(u)}{\int_{u\in\mathbf{U}} \Pr(no_q|u)\pi(u)du}. \tag{4.20}$$

To simplify the notation, we use $\mathbb{E}^{yes_q}$ and $\mathbb{E}^{no_q}$ as posterior expectation operators:

$$\mathbb{E} = \mathbb{E}_{u\sim\pi_v},$$

$$\mathbb{E}^{yes_q} = \mathbb{E}_{u\sim\pi^{yes_q}},$$

$$\mathbb{E}^{no_q} = \mathbb{E}_{u\sim\pi^{no_q}}.$$

Due to the local nature of the bound query $q[v_j^r, b]$, the posterior distribution π^{yes_q} (or π^{no_q}) differs from the prior distribution π only in the dimension of the LVF parameter v_j^r (since, due to the independence assumption stated above, each LVF parameter is independent of the rest). Therefore, the query only affects the expected value of the subutility u_j for factor F_j, and does not affect other factors:

$$\begin{aligned} EU(\mathbf{x}, \pi^{yes_q}) &= \mathbb{E}^{yes_q}[u(\mathbf{x})] \\ &= \sum_{j=1}^{M} \mathbb{E}^{yes_q}[u_j(\mathbf{x}_j)] \\ &= \mathbb{E}^{yes_q}[u_j(\mathbf{x}_j)] + \sum_{j'\neq j} \mathbb{E}^{yes_q}[u_{j'}(\mathbf{x}_{j'})] \\ &= \mathbb{E}^{yes_q}[u_j(\mathbf{x}_j)] + \sum_{j'\neq j} \mathbb{E}[u_{j'}(\mathbf{x}_{j'})]. \end{aligned} \tag{4.21}$$

The expected posterior subutility $\mathbb{E}^{yes_q}[u_j(\mathbf{x}_j)]$ in the equation above can be decomposed further by taking advantage of the fact that a response to a local bound query $q[v_j^r, b]$ affects the expected value of factor F_j subutilities only through the change in the expected value of the query parameter v_j^r (again, because of the independence assumption):

$$\begin{aligned} \mathbb{E}^{yes_{q[v_j^r,b]}}[u_j(\mathbf{x}_j)] &= \lambda_j \sum_{k=1}^{N_j} C^k_{\mathbf{x}_j}\, \mathbb{E}^{yes_q}[v_j^k] \\ &= \lambda_j\, C^r_{\mathbf{x}_j}\, \mathbb{E}^{yes_q}[v_j^r] + \lambda_j \sum_{k\neq r} C^k_{\mathbf{x}_j}\, \mathbb{E}^{yes_q}[v_j^k] \\ &= \lambda_j\, C^r_{\mathbf{x}_j}\, \mathbb{E}^{yes_q}[v_j^r] + \lambda_j \sum_{k\neq r} C^k_{\mathbf{x}_j}\, \mathbb{E}[v_j^k] \\ &= \lambda_j\, C^r_{\mathbf{x}_j}\, \mathbb{E}^{yes_q}[v_j^r] + \Big(\mathbb{E}[u_j(\mathbf{x}_j)] - \lambda_j\, C^r_{\mathbf{x}_j} \mathbb{E}[v_j^r]\Big) \\ &= \mathbb{E}[u_j(\mathbf{x}_j)] + \lambda_j\, C^r_{\mathbf{x}_j}\, \left(\mathbb{E}^{yes_q}[v_j^r] - \mathbb{E}[v_j^r]\right). \end{aligned} \tag{4.22}$$

Thus, the difference between the posterior expectation $\mathbb{E}^{yes_q}[u_j(\mathbf{x}_j)]$ and the prior expectation $\mathbb{E}[u_j(\mathbf{x}_j)]$ is proportional to $\lambda_j\ C^r_{\mathbf{x}_j}\ \left(\mathbb{E}^{yes_q}[v^r_j] - \mathbb{E}[v^r_j]\right)$:

$$\mathbb{E}^{yes_q}[u_j(\mathbf{x}_j)] - \mathbb{E}[u_j(\mathbf{x}_j)] = \lambda_j\ C^r_{\mathbf{x}_j}\ \left(\mathbb{E}^{yes_q}[v^r_j] - \mathbb{E}[v^r_j]\right). \tag{4.23}$$

The response $yes_{q[v^r_j,b]}$ changes $\mathbb{E}[u_j(\mathbf{x}_j)]$ only by changing $\mathbb{E}[v^r_j]$ to $\mathbb{E}^{yes_q}[v^r_j]$). Let $dep(v^r_j)$ be the set of all local outcomes $\mathbf{x}_j$ whose subutility "depends" on the local value parameter v^r_j:

$$dep(v^r_j) = \{\mathbf{x}_j \mid C^r_{\mathbf{x}_j} \neq 0\}. \tag{4.24}$$

If $\mathbf{x}_j \notin dep(v^r_j)$, then a query involving v^r_j will not change the expected value of $u_j(\mathbf{x}_j)$, because the coefficient $C^r_{\mathbf{x}_j}$ is zero (Eq. 4.23):

$$\mathbb{E}^{yes_q}[u_j(\mathbf{x}_j)] = \mathbb{E}[u_j(\mathbf{x}_j)],\ \text{if}\ \mathbf{x}_j \notin dep(v^r_j).$$

Otherwise, if $\mathbf{x}_j \in dep(v^r_j)$, then the expected posterior value of $u_j(\mathbf{x}_j)$ changes only because of the change in the posterior expectation of v^r_j (Eq. 4.22):

$$\mathbb{E}^{yes_{q[v^r_j,b]}}[u_j(\mathbf{x}_j)] = \mathbb{E}[u_j(\mathbf{x}_j)] + \lambda_j\ C^r_{\mathbf{x}_j}\ \left(\mathbb{E}^{yes_q}[v^r_j] - \mathbb{E}[v^r_j]\right).$$

By substituting the expression for $\mathbb{E}^{yes_q}[u_j(\mathbf{x}_j)]$ in Eq. 4.22 into Eq. 4.21, we obtain the final equation for the expected utility of the posterior belief state π^{yes_q}:

$$\begin{aligned}\mathbb{E}^{yes_{q[v^r_j,b]}}[u(\mathbf{x})] &= \mathbb{E}^{yes_q}[u_j(\mathbf{x}_j)] + \sum_{j'\neq j}\mathbb{E}[u_{j'}(\mathbf{x}_{j'})] \qquad (4.25)\\ &= \begin{cases} \mathbb{E}[u_j(\mathbf{x}_j)] + \sum_{j'\neq j}\mathbb{E}[u_{j'}(\mathbf{x}_{j'})],\ \text{if}\ \mathbf{x}_j \notin dep(v^r_j), \\ \mathbb{E}[u_j(\mathbf{x}_j)] + \lambda_j\ C^r_{\mathbf{x}_j}\ \left(\mathbb{E}^{yes_q}[v^r_j] - \mathbb{E}[v^r_j]\right) + \sum_{j'\neq j}\mathbb{E}[u_{j'}(\mathbf{x}_{j'})],\ \text{otherwise.}\end{cases}\end{aligned}$$

An analogous expression holds for $\mathbb{E}^{no_{q[v^r_j,b]}}[u(\mathbf{x})]$.

Our goal in this subsection is to compute the values (MEUs) of posterior belief states π^{yes_q} and π^{no_q} that result from responses to an LBQ $q[v^r_j, b]$. The maximum expected utility of π^{yes_q} is the expected utility of the best outcome for that belief state. We first introduce the function

$\rho_j(\mathbf{x}_j)$ that, for a given instantiation $\mathbf{x}_j$ in factor F_j, specifies the best possible expected value of subutilities in other factors:

$$\rho_j(\mathbf{x}_j) = \max_{\mathbf{x}' \text{ s.t. } \mathbf{x}'_j = \mathbf{x}_j} \sum_{j' \neq j} \mathbb{E}[u_{j'}(\mathbf{x}'_{j'})]. \tag{4.26}$$

Using Eq. 4.25, we obtain:

$$\begin{aligned} MEU(\pi^{yes_{q[v_j^r,b]}}) &= \max_{\mathbf{x}} \mathbb{E}^{yes_{q[v_j^r,b]}}[u(\mathbf{x})] \\ &= \max_{\mathbf{x}_j} \left[\mathbb{E}^{yes_q}[u_j(\mathbf{x}_j)] + \max_{\mathbf{x}' \text{ s.t. } \mathbf{x}'_j = \mathbf{x}_j} \sum_{j' \neq j} \mathbb{E}[u_{j'}(\mathbf{x}'_{j'})] \right] \\ &= \max_{\mathbf{x}_j} \left[\mathbb{E}^{yes_q}[u_j(\mathbf{x}_j)] + \rho_j(\mathbf{x}_j) \right] \\ &= \max \begin{cases} \max_{\mathbf{x}_j \notin dep(v_j^r)} \left[\mathbb{E}[u_j(\mathbf{x}_j)] + \rho_j(\mathbf{x}_j) \right], \\ \max_{\mathbf{x}_j \in dep(v_j^r)} \left[\lambda_j \, C^r_{\mathbf{x}_j} \, \mathbb{E}^{yes_q}[v_j^r] + \left(\mathbb{E}[u_j(\mathbf{x}_j)] - \lambda_j \, C^r_{\mathbf{x}_j} \mathbb{E}[v_j^r] \right) + \rho_j(\mathbf{x}_j) \right]. \end{cases} \end{aligned} \tag{4.27}$$

Thus, to evaluate one LBQ $q[v_j^r, b]$, we need to consider all local configurations $\mathbf{x}_j$ that do not "depend" on v_j^r (the number of such configurations is bounded by N_j, the total number of configurations in factor F_j), and all local configurations that do depend on v_j^r (the number of such configurations is determined by the GAI structure and is typically very small). Assuming that the function $\rho_j(\mathbf{x}_j)$ is precomputed, and that evaluating the posterior $\mathbb{E}^{yes_q}[v_j^r]$ is not hard (i.e., the running time is bounded by some constant, though this depends on the form of the density representation), the evaluation of $MEU(\pi^{yes_{q[v_j^r,b]}})$ is linear in the number of local configurations in factor F_j (or, equivalently, exponential in the number of attributes in F_j).

The MEU of π^{no_q} can be computed in the same way, by replacing yes_q with no_q.

4.2.3.2 Expected posterior utility of a local bound query

The expected posterior utility of an LBQ $q[v_j^r, b]$ is

$$EPU(q[v_j^r, b], \pi) = \Pr(yes_q|\pi) \max_{\mathbf{x}} \mathbb{E}^{yes_q}[u(\mathbf{x})] + \Pr(no_q|\pi) \max_{\mathbf{x}} \mathbb{E}^{no_q}[u(\mathbf{x})]. \tag{4.28}$$

By substituting Eq. 4.27 into Eq. 4.28, we obtain the following expression for the EPU of a local bound query:

$$
\begin{aligned}
EPU(q[v_j^r, b], \pi) &= \Pr(yes_q|\pi) \max_{\mathbf{x}} \mathbb{E}^{yes_q}[u(\mathbf{x})] + \Pr(no_q|\pi) \max_{\mathbf{x}} \mathbb{E}^{no_q}[u(\mathbf{x})] \\
&= \Pr(yes_q|\pi) \ \max \left\{ \begin{array}{l} \max_{\mathbf{x}_j \notin dep(v_j^r)} \left[\mathbb{E}[u_j(\mathbf{x}_j)] + \rho_j(\mathbf{x}_j)\right] \\ \max_{\mathbf{x}_j \in dep(v_j^r)} \left[\lambda_j \ C^r_{\mathbf{x}_j} \ \mathbb{E}^{yes_q}[v_j^r] + \Big(\mathbb{E}[u_j(\mathbf{x}_j)] - \lambda_j \ C^r_{\mathbf{x}_j} \mathbb{E}[v_j^r] + \rho_j(\mathbf{x}_j)\Big)\right] \end{array} \right. \\
&\quad + \Pr(no_q|\pi) \ \max \left\{ \begin{array}{l} \max_{\mathbf{x}_j \notin dep(v_j^r)} \left[\mathbb{E}[u_j(\mathbf{x}_j)] + \rho_j(\mathbf{x}_j)\right] \\ \max_{\mathbf{x}_j \in dep(v_j^r)} \left[\lambda_j \ C^r_{\mathbf{x}_j} \ \mathbb{E}^{no_q}[v_j^r] + \Big(\mathbb{E}[u_j(\mathbf{x}_j)] - \lambda_j \ C^r_{\mathbf{x}_j} \mathbb{E}[v_j^r] + \rho_j(\mathbf{x}_j)\Big)\right] \end{array} \right. \\
&= \Pr(yes_q|\pi) \ \max \left\{ \begin{array}{l} \gamma(j, r) \\ \max_{\mathbf{x}_j \in dep(v_j^r)} \left[d_1(\mathbf{x}_j|j, r) \ \mu^{yes}(j, r, b) + d_2(\mathbf{x}_j|j, r)\right] \end{array} \right. \\
&\quad + \Pr(no_q|\pi) \ \max \left\{ \begin{array}{l} \gamma(j, r) \\ \max_{\mathbf{x}_j \in dep(v_j^r)} \left[d_1(\mathbf{x}_j|j, r) \ \mu^{no}(j, r, b) + d_2(\mathbf{x}_j|j, r)\right], \end{array} \right.
\end{aligned}
\tag{4.29}
$$

where

$$
\begin{aligned}
\gamma(j, r) &= \max_{\mathbf{x}_j \notin dep(v_j^r)} \mathbb{E}[u_j(\mathbf{x}_j)] + \rho_j(\mathbf{x}_j), \\
\mu^{yes}(j, r, b) &= \mathbb{E}^{yes_q}[v_j^r], \\
\mu^{no}(j, r, b) &= \mathbb{E}^{no_q}[v_j^r], \\
d_1(\mathbf{x}_j|j, r) &= \lambda_j \ C^r_{\mathbf{x}_j}, \\
d_2(\mathbf{x}_j|j, r) &= \mathbb{E}[u_j(\mathbf{x}_j)] - \lambda_j \ C^r_{\mathbf{x}_j} \mathbb{E}[v_j^r] + \rho_j(\mathbf{x}_j).
\end{aligned}
$$

We can use Eq. 4.29 to evaluate LBQs in the query set Q. The myopically optimal query q^* is the one with the highest EPU:

$$
q^* = q[v_{j*}^{r*}, b^*] = \arg\max_{j,r,b} EPU(q[v_j^r, b], \pi). \tag{4.30}
$$

Because the bound parameter b is continuous, we cannot find q^* by simply enumerating all the queries in Q. However, in certain settings, we can compute the optimal bound b *for a specific*

local value parameter v_j^r. Since the number of local value parameters v_j^k is proportional to the size of the GAI utility representation (we have N_j local configurations for each factor F_j), in such cases it might be feasible to find the optimal query by enumerating LVF parameters and computing optimal bounds for each of them. In the following subsection, we show that this is feasible when the density π is a mixture of uniform probability distributions.

4.2.3.3 Mixtures of uniforms and bound queries

The prior π over local value parameters can assume many different forms, both parametric and non-parametric. The density π would ideally facilitate several functions, including 1) accurate representation of the systems' beliefs about user utilities; 2) accurate and tractable updating of the belief state after acquisition of new preference information; and, 3) support for query optimization. Because LVF parameters are continuous, the parametric distributions that have been proposed in the literature include a Gaussian distribution (Chajewska et al., 2000), a mixture of Gaussians (Chajewska et al., 2000), a mixture of truncated Gaussians (Boutilier, 2002), and a mixture of uniforms (Boutilier, 2002; Braziunas and Boutilier, 2005). Mixture distributions are popular because they can compensate for the inflexibility of parametric models. Their representational accuracy can be increased by adding more mixture components; in this sense, all the mixture distributions offer similar advantages in representing user utilities. With respect to belief state updates, none of the parametric distributions are closed under responses to comparison queries, and only mixtures of truncated Gaussians and uniforms are closed under responses to bound queries. The common way of dealing with the belief state maintenance problem is to sample from the posterior distribution and refit the original parametric distribution using a standard method, such as EM (Chajewska et al., 2000). Of course, this results in additional computational burden and the loss of representational accuracy with further updates. Finally, some densities are better than others in facilitating query optimization. Densities that are not closed under updates are especially difficult to deal with when optimizing for the best query, since calculation of each query's value involves estimating the values of all potential posterior

belief states. Even for bound queries, which result in closed updates of mixtures of truncated Gaussians and uniforms, only mixtures of uniform densities support analytical (myopic) optimization of the bound query's continuous bound parameter (which is what we demonstrate next). Therefore, in the remainder of this chapter, we use mixtures of uniforms to develop a tractable myopic elicitation algorithm.

In previous work, Boutilier (2002) used mixtures of uniforms over unstructured (flat) utility models. Here, we extend the framework to GAI utilities and show how to maintain the belief state and compute optimal LBQ bounds $b^*_{v^r_j}$ for every LVF parameter v^r_j. Specifying prior information over local utility parameters as a mixture of uniform distributions offers several advantages for utility elicitation. With enough components, a mixture of uniforms is flexible enough to approximate many standard distributions; furthermore, it also fits nicely with the type of bound queries we consider here. Because the posterior distribution after a response to a query remains a mixture of uniforms (we only need to update the component weights and bounds), it is possible to maintain an exact density over utility parameters throughout the elicitation process. Most importantly, with prior densities over utility parameters expressed as mixtures of uniform distributions, we can calculate the optimal LBQ bounds $b^*_{v^r_j}$ analytically.

Let $\pi_{v^r_j}$ be the marginal probability density over the local value parameter v^r_j. We model the density over local value parameters as a mixture of K *multidimensional* uniform distributions. This implies a strong assumption of *independence* among the random variables representing local value parameters. Instead of working with multidimensional uniform distributions, the independence condition allows us to work with marginal probability distributions $\pi_{v^r_j}$, one for each local value parameter.

Each probabability distribution $\pi_{v^r_j}$ is a mixture of (one-dimensional) uniform distributions. Such mixture is denoted by K triples $\langle w_k, \alpha_k, \beta_k \rangle$, where α_k and β_k are lower and upper bounds of the k^{th} component, and w_k is its weight ($w_k \in [0, 1]$ for all k, and $\sum_k w_k = 1$). Figure 4.6 shows four examples of mixtures of uniform distributions.

In this subsection, we label a generic value parameter v_j^r as v, and use π to denote $\pi_{v_j^r}$.

The mean value of $\pi = \langle w_k, \alpha_k, \beta_k \rangle, k = 1..K$ is

$$\mu = \mathbb{E}_{v\sim\pi}[v] = \sum_{k=1}^{K} w_k \frac{\alpha_k + \beta_k}{2}. \tag{4.31}$$

Let $q[v, b]$ be a local bound query with responses yes_b and no_b; the response yes_b indicates that the local value parameter v is above the bound b, and response no_b indicates that the local value parameter v is below the bound b. We assume a noiseless response model:

$$\Pr(yes_b|v) = \begin{cases} 1, \text{ if } v \geq b, \\ 0, \text{otherwise.} \end{cases}$$

$$\Pr(no_b|v) = 1 - \Pr(yes_b|v).$$

If v is distributed according to π, the probability of response yes_b (i.e., the probability that v is greater than b) is:

$$\Pr(yes_q|\pi) = \sum_{k=1}^{K} w_k \frac{\beta_k - \max(\alpha_k, \min(\beta_k, b))}{\beta_k - \alpha_k} = \sum_{k=1}^{K} w_k \frac{\beta_k - b_k(b)}{\beta_k - \alpha_k}, \tag{4.32}$$

where

$$b_k(b) = \max(\alpha_k, \min(\beta_k, b)) = \begin{cases} \alpha_k, \text{if } b \leq \alpha_k, \\ b, \text{if } \alpha_k < b < \beta_k, \\ \beta_k, \text{if } b \geq \beta_k. \end{cases}$$

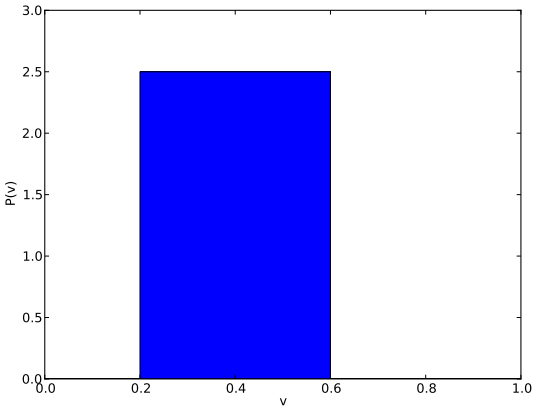
Similarly,

$$\Pr(no_b|\pi) = \sum_{k=1}^{K} w_k \frac{b_k(b) - \alpha_k}{\beta_k - \alpha_k}. \tag{4.33}$$

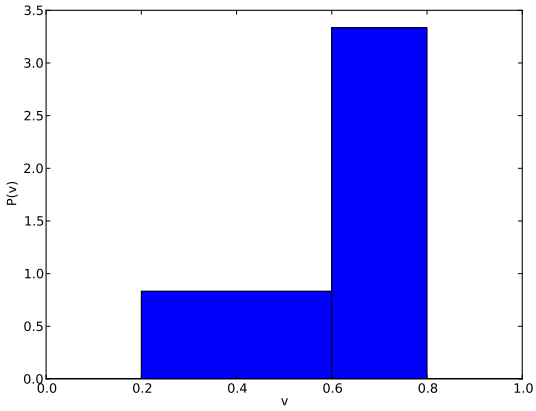
After receiving response yes_b, the lower bounds α_k of each mixture component are updated to $b_k(b) = \max(\alpha_k, \min(\beta_k, b))$. Similarly, the response no_b affects upper bounds of mixture components. The posterior weights $w_k^{yes_b}$ and $w_k^{no_b}$ are:

$$w_k^{yes_b} = \frac{w_k \frac{\beta_k - b_k(b)}{\beta_k - \alpha_k}}{\sum_{k'} w_{k'} \frac{\beta_{k'} - b_{k'}(b)}{\beta_{k'} - \alpha_{k'}}},$$

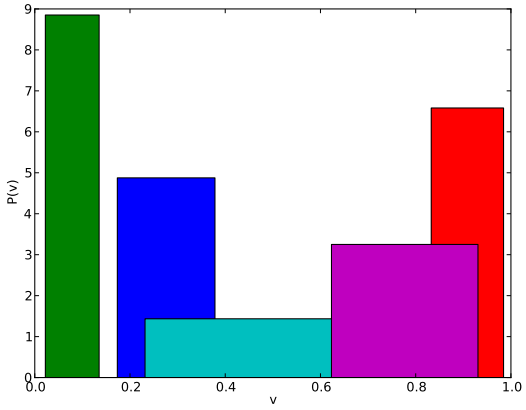
$$w_k^{no_b} = \frac{w_k \frac{b_k(b) - \alpha_k}{\beta_k - \alpha_k}}{\sum_{k'} w_{k'} \frac{b_{k'}(b) - \alpha_{k'}}{\beta_{k'} - \alpha_{k'}}}.$$



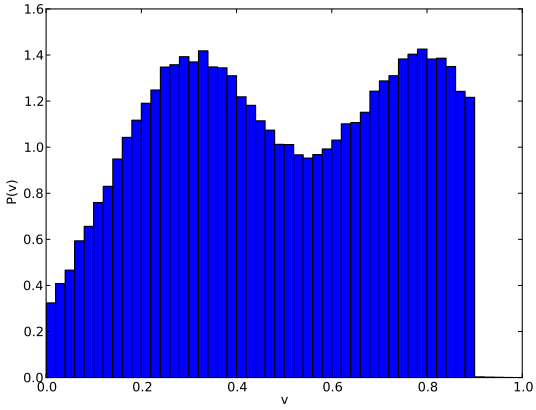
(a) Uniform distribution



(b) Mixture of 2 uniforms



(c) Mixture of 5 uniforms



(d) Mixture of 50 uniforms

Figure 4.6: Four examples of mixtures of uniform distributions. Part (a) shows the density of a uniform distribution with bounds $\alpha = 0.2$, $\beta = 0.6$. Part (b) is a mixture of two uniforms. The first component's bounds are $\alpha_1 = 0.2$, $\beta_1 = 0.6$, and its weight is $w_1 = 0.33$; the second component's bounds are $\alpha_2 = 0.6$, $\beta_2 = 0.8$, and its weight is $w_2 = 0.67$. Part (c) is a mixture of five components, some of which "overlap" each other. Part (d) shows a mixture of 50 components that approximates a bimodal distribution (a truncated mixture of two Gaussians).

The posterior distributions π^{yes_b} and π^{no_b} reflect updated mixture bounds and weights:

$$\pi^{yes_b} = \langle w_k^{yes_b}, b_k(b), \beta_k \rangle, k = 1..K,$$
$$\pi^{no_b} = \langle w_k^{no_b}, b_k(b), \beta_k \rangle, k = 1..K. \tag{4.34}$$

The posterior means μ^{yes_b} and μ^{no_b} are:

$$\mu^{yes_b} = \sum_{k=1}^{K} w_k^{yes_b} \ \frac{b_k(b) + \beta_k}{2},$$
$$\mu^{no_b} = \sum_{k=1}^{K} w_k^{no_b} \ \frac{\alpha_k + b_k(b)}{2}. \tag{4.35}$$

Optimization of query bound b

Our ultimate goal is to find the bound b that maximizes the EPU of a bound query on a specific parameter v_j^r (Eq. 4.29). By looking at the functional form of expressions in Eq. 4.29 (and, for now, ignoring the max operator), we can rewrite Eq. 4.29 as a function of the bound b:

$$f(b|\pi) = \Pr(yes_b|\pi) \ (c_1^{yes} \ \mu^{yes_b} + c_2^{yes}) + \Pr(no_b|\pi) \ (c_1^{no} \ \mu^{no_b} + c_2^{no}), \tag{4.36}$$

where $c_1^{yes}, c_2^{yes}, c_1^{no}, c_2^{no}$ are coefficients that do not depend on the bound b.

By substituting quantities from Eq. 4.32 and Eq. 4.35 into the first part of the expression above, we obtain

$$\begin{aligned}
&\Pr(yes_b|\pi) \ (c_1^{yes} \ \mu^{yes_b} + c_2^{yes}) = \\
&= \left[\sum_{k=1}^{K} w_k \ \frac{\beta_k - b_k(b)}{\beta_k - \alpha_k}\right] \left[c_1^{yes} \ \left(\sum_{k=1}^{K} \frac{w_k \ \frac{\beta_k - b_k(b)}{\beta_k - \alpha_k}}{\sum_{k'} w_{k'} \ \frac{\beta_{k'} - b_{k'}(b)}{\beta_{k'} - \alpha_{k'}}} \ \frac{b_k(b) + \beta_k}{2}\right) + c_2^{yes}\right] \\
&= \left[\sum_{k=1}^{K} w_k \ \frac{\beta_k - b_k(b)}{\beta_k - \alpha_k}\right] \left[\frac{c_1^{yes}}{\sum_{k'} w_{k'} \ \frac{\beta_{k'} - b_{k'}(b)}{\beta_{k'} - \alpha_{k'}}} \left(\sum_{k=1}^{K} w_k \ \frac{\beta_k - b_k(b)}{\beta_k - \alpha_k} \ \frac{b_k(b) + \beta_k}{2}\right) + c_2^{yes}\right] \\
&= c_1^{yes} \ \left(\sum_{k=1}^{K} w_k \ \frac{\beta_k - b_k(b)}{\beta_k - \alpha_k} \ \frac{b_k(b) + \beta_k}{2}\right) + c_2^{yes} \ \left(\sum_{k=1}^{K} w_k \ \frac{\beta_k - b_k(b)}{\beta_k - \alpha_k}\right).
\end{aligned}$$

Similarly,

$$Pr(no_b|\pi) \ (c_1^{no} \ \mu^{no_b} + c_2^{no}) = c_1^{no} \ \left(\sum_{k=1}^{K} w_k \ \frac{b_k(b) - \alpha_k}{\beta_k - \alpha_k} \ \frac{b_k(b) + \alpha_k}{2}\right) + c_2^{no} \ \left(\sum_{k=1}^{K} w_k \ \frac{b_k(b) - \alpha_k}{\beta_k - \alpha_k}\right).$$

By adding the two parts together, the function $f(b|\pi)$ from Eq. 4.36 becomes

$$\begin{aligned}
f(b|\pi) &= \Pr(yes_b|\pi)\ (c_1^{yes}\ \mu^{yes_b} + c_2^{yes}) + \Pr(no_b|\pi)\ (c_1^{no}\ \mu^{no_b} + c_2^{no}) = \\
&= c_1^{yes}\ \left(\sum_{k=1}^{K} w_k\ \frac{\beta_k - b_k(b)}{\beta_k - \alpha_k}\ \frac{b_k(b) + \beta_k}{2}\right) + c_2^{yes}\ \left(\sum_{k=1}^{K} w_k\ \frac{\beta_k - b_k(b)}{\beta_k - \alpha_k}\right) + \\
&\quad c_1^{no}\ \left(\sum_{k=1}^{K} w_k\ \frac{b_k(b) - \alpha_k}{\beta_k - \alpha_k}\ \frac{b_k(b) + \alpha_k}{2}\right) + c_2^{no}\ \left(\sum_{k=1}^{K} w_k\ \frac{b_k(b) - \alpha_k}{\beta_k - \alpha_k}\right) \\
&= \sum_{k=1}^{K}\left[c_1^{yes}\ w_k\ \frac{\beta_k - b_k(b)}{\beta_k - \alpha_k}\ \frac{b_k(b) + \beta_k}{2} + c_2^{yes}\ w_k\ \frac{\beta_k - b_k(b)}{\beta_k - \alpha_k} + \right. \\
&\quad \left. c_1^{no}\ w_k\ \frac{b_k(b) - \alpha_k}{\beta_k - \alpha_k}\ \frac{b_k(b) + \alpha_k}{2} + c_2^{no}\ w_k\ \frac{b_k(b) - \alpha_k}{\beta_k - \alpha_k}\right]. \qquad (4.37)
\end{aligned}$$

For a given query bound b, the set $\{1, \ldots, K\}$ of mixture component indices can be partitioned into three sets K^α, K^b, and K^β, depending on whether the query bound b is below the lower bound of the component (K^α), within the bounds of the component (K^b), or above the upper bound of the component (K^β):

$$K^\alpha = \{k \mid b \leq \alpha_k\}, \qquad (4.38)$$

$$K^b = \{k \mid \alpha_k < b < \beta_k\}, \qquad (4.39)$$

$$K^\beta = \{k \mid b \geq \beta_k\}. \qquad (4.40)$$

For $k \in K^\alpha$, $b_k(b) = \alpha_k$ (Eq. 4.32), and therefore the term inside the sum in Eq. 4.37 simplifies to

$$c_1^{yes}\ w_k\ \frac{\beta_k - b_k(b)}{\beta_k - \alpha_k}\ \frac{b_k(b) + \beta_k}{2} + c_2^{yes}\ w_k\ \frac{\beta_k - b_k(b)}{\beta_k - \alpha_k} + c_1^{no}\ w_k\ \frac{b_k(b) - \alpha_k}{\beta_k - \alpha_k}\ \frac{b_k(b) + \alpha_k}{2} + c_2^{no}\ w_k\ \frac{b_k(b) - \alpha_k}{\beta_k - \alpha_k}$$
$$= w_k\ (c_1^{yes}\mu_k + c_2^{yes}).$$

Similarly, for $k \in K^\beta$, $b_k(b) = \beta_k$. Therefore,

$$c_1^{yes}\ w_k\ \frac{\beta_k - b_k(b)}{\beta_k - \alpha_k}\ \frac{b_k(b) + \beta_k}{2} + c_2^{yes}\ w_k\ \frac{\beta_k - b_k(b)}{\beta_k - \alpha_k} + c_1^{no}\ w_k\ \frac{b_k(b) - \alpha_k}{\beta_k - \alpha_k}\ \frac{b_k(b) + \alpha_k}{2} + c_2^{no}\ w_k\ \frac{b_k(b) - \alpha_k}{\beta_k - \alpha_k}$$
$$= w_k\ (c_1^{no}\mu_k + c_2^{no}).$$

Finally, for $k \in K^b$, $b_k(b) = b$, so the term becomes a quadratic function of b:

$$c_1^{yes}\ w_k\ \frac{\beta_k - b_k(b)}{\beta_k - \alpha_k}\ \frac{b_k(b) + \beta_k}{2} + c_2^{yes}\ w_k\ \frac{\beta_k - b_k(b)}{\beta_k - \alpha_k} + c_1^{no}\ w_k\ \frac{b_k(b) - \alpha_k}{\beta_k - \alpha_k}\ \frac{b_k(b) + \alpha_k}{2} + c_2^{no}\ w_k\ \frac{b_k(b) - \alpha_k}{\beta_k - \alpha_k}$$
$$= \frac{w_k}{\beta_k - \alpha_k}\ \left[\frac{c_1^{yes}}{2}(\beta_k^2 - b^2) + c_2^{yes}(\beta_k - b) + \frac{c_1^{no}}{2}(b^2 - \alpha_k^2) + c_2^{no}(b - \alpha_k)\right]$$
$$= \frac{w_k}{\beta_k - \alpha_k}\ \left[\frac{(c_1^{no} - c_1^{yes})}{2}\ b^2 + (c_2^{no} - c_2^{yes})\ b + \left(\frac{c_1^{yes}}{2}\beta_k^2 - \frac{c_1^{no}}{2}\alpha_k^2 + c_2^{yes}\beta_k - c_2^{no}\alpha_k\right)\right].$$

We can now rewrite Eq. 4.36 as follows:

$$f(b|\pi) = \Pr(yes_b|\pi)\ (c_1^{yes}\ \mu^{yes_b} + c_2^{yes}) + \Pr(no_b|\pi)\ (c_1^{no}\ \mu^{no_b} + c_2^{no}) =$$
$$= \sum_{k=1}^{K}\left[c_1^{yes}\ w_k\ \frac{\beta_k - b_k(b)}{\beta_k - \alpha_k}\ \frac{b_k(b) + \beta_k}{2} + c_2^{yes}\ w_k\ \frac{\beta_k - b_k(b)}{\beta_k - \alpha_k} + \right.$$
$$\left. c_1^{no}\ w_k\ \frac{b_k(b) - \alpha_k}{\beta_k - \alpha_k}\ \frac{b_k(b) + \alpha_k}{2} + c_2^{no}\ w_k\ \frac{b_k(b) - \alpha_k}{\beta_k - \alpha_k}\right] =$$

$$= \sum_{k \in K^\alpha} w_k\ (c_1^{yes}\mu_k + c_2^{yes}) + \sum_{k \in K^\beta} w_k\ (c_1^{no}\mu_k + c_2^{no}) +$$
$$\sum_{k \in K^b} \frac{w_k}{\beta_k - \alpha_k}\ \left[\frac{(c_1^{no} - c_1^{yes})}{2}\ b^2 + (c_2^{no} - c_2^{yes})\ b + \left(\frac{c_1^{yes}}{2}\beta_k^2 - \frac{c_1^{no}}{2}\alpha_k^2 + c_2^{yes}\beta_k - c_2^{no}\alpha_k\right)\right] =$$

$$= \left[\frac{(c_1^{no} - c_1^{yes})}{2}\ b^2 + (c_2^{no} - c_2^{yes})\ b\right] \sum_{k \in K^b} \frac{w_k}{\beta_k - \alpha_k} +$$
$$\sum_{k \in K^b} \frac{w_k(c_1^{yes}\beta_k^2 - c_1^{no}\alpha_k^2 + 2c_2^{yes}\beta_k - 2c_2^{no}\alpha_k)}{2(\beta_k - \alpha_k)} + \sum_{k \in K^\alpha} w_k\ (c_1^{yes}\mu_k + c_2^{yes}) + \sum_{k \in K^\beta} w_k\ (c_1^{no}\mu_k + c_2^{no}).$$

This last form of $f(b|\pi)$ provides good insight into optimization with mixtures of uniforms. We can see that no matter how we partition $\{1, \ldots, K\}$ into three sets K^α, K^b, and K^β, the derivative of $f(b|\pi)$ (with respect to b) is $(c_1^{no} - c_1^{yes})\ b + (c_2^{no} - c_2^{yes})$, and therefore the optimum point of $f(b|\pi) = \Pr(yes_b|\pi)\ (c_1^{yes}\ \mu^{yes_b} + c_2^{yes}) + \Pr(no_b|\pi)\ (c_1^{no}\ \mu^{no_b} + c_2^{no})$ always occurs at

$$b^* = \frac{c_2^{yes} - c_2^{no}}{c_1^{no} - c_1^{yes}}. \tag{4.41}$$

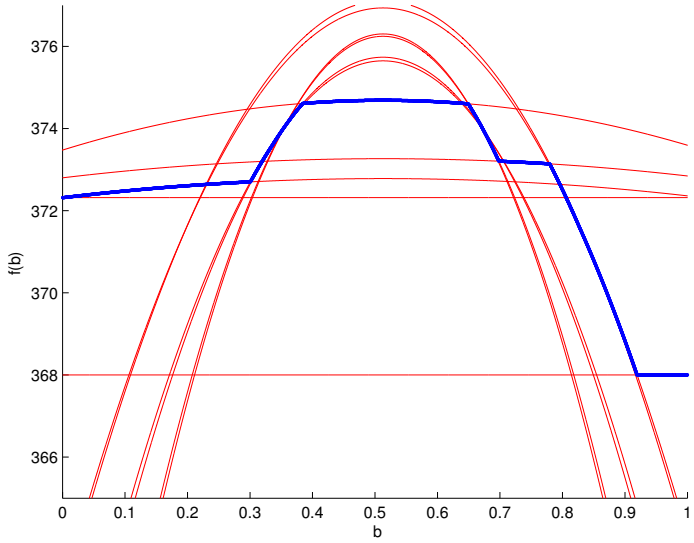


Figure 4.7: An example of a piecewise quadratic function of b (in bold/blue), with a common maximum for all components (in red). The EPU of a bound query is of the same form with respect to the bound parameter b.

Since the partitioning of mixture components depends on the value of b, the function is a piecewise quadratic in b (see Fig. 4.7 for an example).

For LBQs, the query bound b is restricted to the $[0, 1]$ interval. The following cases apply:

- If $c_1^{no} - c_1^{yes} > 0$, then the piecewise quadratic function is convex, and b^* is a minimum point. Thus, the maximum occurs at either $b = 0$ or $b = 1$.

- Otherwise, if $c_1^{no} - c_1^{yes} \leq 0$, then the function is concave, and the maximum occurs at b^*. However, if b^* is outside $[0, 1]$, then the maximum is either $b = 0$ or $b = 1$.

If the maximum occurs at either $b = 0$ or $b = 1$ (or outside the $[0, 1]$ interval), the local query provides no value. Indeed, when $b = 0$, the equation simplifies to $c_1^{yes}\mu + c_2^{yes}$, and when $b = 1$, the equation simplifies to $c_1^{no}\mu + c_2^{no}$. The query has value of information only if $c_1^{no} - c_1^{yes} \leq 0$ (the function is concave) and $b^* = \frac{c_2^{yes} - c_2^{no}}{c_1^{no} - c_1^{yes}}$ is within $[0, 1]$.

Maximal EPU of local bound query on specific LVF parameter

Let's recall that the expected posterior utility of the local bound query $q[v_j^r, b]$ is (Eq. 4.29)

$$EPU(q[v_j^r, b], \pi) = \Pr(yes_q|\pi) \max_{\mathbf{x}} \mathbb{E}^{yes_q}[u(\mathbf{x})] + \Pr(no_q|\pi) \max_{\mathbf{x}} \mathbb{E}^{no_q}[u(\mathbf{x})]$$

$$= \Pr(yes_q|\pi) \ \max \begin{cases} \gamma(j,r) \\ \max_{\mathbf{x}_j \in dep(v_j^r)} \left[d_1(\mathbf{x}_j|j,r) \ \mu^{yes}(j,r,b) + d_2(\mathbf{x}_j|j,r) \right] \end{cases}$$

$$+ \Pr(no_q|\pi) \ \max \begin{cases} \gamma(j,r) \\ \max_{\mathbf{x}_j \in dep(v_j^r)} \left[d_1(\mathbf{x}_j|j,r) \ \mu^{no}(j,r,b) + d_2(\mathbf{x}_j|j,r) \right], \end{cases}$$

where

$$\gamma(j,r) = \max_{\mathbf{x}_j \notin dep(v_j^r)} \mathbb{E}[u_j(\mathbf{x}_j)] + \rho_j(\mathbf{x}_j),$$

$$\mu^{yes}(j,r,b) = \mathbb{E}^{yes_q}[v_j^r],$$

$$\mu^{no}(j,r,b) = \mathbb{E}^{no_q}[v_j^r],$$

$$d_1(\mathbf{x}_j|j,r) = \lambda_j \ C^r_{\mathbf{x}_j},$$

$$d_2(\mathbf{x}_j|j,r) = \mathbb{E}[u_j(\mathbf{x}_j)] - \lambda_j \ C^r_{\mathbf{x}_j} \mathbb{E}[v_j^r] + \rho_j(\mathbf{x}_j).$$

Due to the structure of GAI utilities, the coefficients $C^r_{\mathbf{x}_j}$, and, therefore, $d_1(\mathbf{x}_j|q[v_j^r, b])$, are always the same for all $\mathbf{x}_j \in dep(v_j^r)$ (i.e., all the non-zero entries in the r^{th} column of the $\mathbf{C}_j$ structure matrix are the same). By comparing the EPU equation above to our piecewise quadratic function $f(b|\pi)$ in Eq. 4.36, we notice that equality of coefficients $d_1(\mathbf{x}_j|q[v_j^r, b])$ for all $\mathbf{x}_j \in dep(v_j^r)$ in both the $\Pr(yes|q,\pi)$ and $\Pr(no|q,\pi)$ parts of the EPU equation means that $c_1^{yes} = c_1^{no}$ for all $\mathbf{x}_j \in dep(v_j^r)$, and, therefore, there is no b that maximizes the EPU in such a case.

Hence, the EPU equation 4.29 reduces to solving

$$\max \begin{cases} \max_{\mathbf{x}_j \in dep(v_j^r)} \Pr(yes_q|\pi)\ \gamma(j,r) + \Pr(no_q|\pi)\ (d_1(\mathbf{x}_j|j,r)\ \mu^{no}(j,r,b) + d_2(\mathbf{x}_j|j,r)) \\ \max_{\mathbf{x}_j \in dep(v_j^r)} \Pr(yes_q|\pi)\ (d_1(\mathbf{x}_j|j,r)\ \mu^{yes}(j,r,b) + d_2(\mathbf{x}_j|j,r)) + \Pr(no_q|\pi)\ \gamma(j,r) \end{cases}$$

$$= \max \begin{cases} \max_{\mathbf{x}_j \in dep(v_j^r)} f_1(b|\mathbf{x}_j,\pi) \\ \max_{\mathbf{x}_j \in dep(v_j^r)} f_2(b|\mathbf{x}_j,\pi), \end{cases} \tag{4.42}$$

where

$$f_1(b|\mathbf{x}_j,\pi) = \Pr(yes_q|\pi)\ \gamma(j,r) + \Pr(no_q|\pi)\ (d_1(\mathbf{x}_j|j,r)\ \mu^{no}(j,r,b) + d_2(\mathbf{x}_j|j,r)),$$
$$f_2(b|\mathbf{x}_j,\pi) = \Pr(yes_q|\pi)\ (d_1(\mathbf{x}_j|j,r)\ \mu^{yes}(j,r,b) + d_2(\mathbf{x}_j|j,r)) + \Pr(no_q|\pi)\ \gamma(j,r).$$

Using Eq. 4.41, we can easily compute the bound $b^*_{\mathbf{x}_j}$ that achieves the optimal value of $f_1(b|\mathbf{x}_j,\pi)$ (for a particular $\mathbf{x}_j \in dep(v_j^r)$). Since $c_1^{yes} = 0, c_2^{yes} = \gamma(j,r), c_1^{no} = d_1(\mathbf{x}_j|j,r), c_2^{no} = d_2(\mathbf{x}_j|j,r)$, the optimal bound is

$$b^*_{\mathbf{x}_j} = \frac{c_2^{yes} - c_2^{no}}{c_1^{no} - c_1^{yes}} = \frac{\gamma(j,r) - d_2(\mathbf{x}_j|j,r)}{d_1(\mathbf{x}_j|j,r)}.$$

Similarly, for $f_2(b|\mathbf{x}_j,\pi)$, $c_1^{yes} = d_1(\mathbf{x}_j|j,r), c_2^{yes} = d_2(\mathbf{x}_j|j,r), c_1^{no} = 0, c_2^{no} = \gamma(j,r)$. For a given $\mathbf{x}_j \in dep(v_j^r)$, the optimal bound $b^*_{\mathbf{x}_j}$ is the same:

$$b^*_{\mathbf{x}_j} = \frac{c_2^{yes} - c_2^{no}}{c_1^{no} - c_1^{yes}} = \frac{d_2(\mathbf{x}_j|j,r) - \gamma}{-d_1(\mathbf{x}_j|j,r)} = \frac{\gamma - d_2(\mathbf{x}_j|j,r)}{d_1(\mathbf{x}_j|j,r)}.$$

Thus, $\max_b f_1(b|\mathbf{x}_j,\pi) = f_1(b^*_{\mathbf{x}_j}|\mathbf{x}_j,\pi)$, and $\max_b f_2(b|\mathbf{x}_j,\pi) = f_2(b^*_{\mathbf{x}_j}|\mathbf{x}_j,\pi)$.

We can now write the expression for the optimal bound for a local bound query on a specific

LVF parameter. For an LBQ on the value parameter v_j^r, the optimal bound b^* is

$$\begin{aligned} b^* &= \operatorname*{argmax}_b EPU(q[v_j^r, b], \pi) \\ &= \operatorname*{argmax}_b \max \begin{cases} \max_{\mathbf{x}_j \in dep(v_j^r)} \max_b f_1(b|\mathbf{x}_j, \pi) \\ \max_{\mathbf{x}_j \in dep(v_j^r)} \max_b f_2(b|\mathbf{x}_j, \pi) \end{cases} \\ &= \operatorname*{argmax}_{b^*_{\mathbf{x}_j},\, \mathbf{x}_j \in dep(v_j^r)} \max \begin{cases} f_1(b^*_{\mathbf{x}_j}|\mathbf{x}_j, \pi) \\ f_2(b^*_{\mathbf{x}_j}|\mathbf{x}_j, \pi), \end{cases} \end{aligned} \tag{4.43}$$

where

$$f_1(b|\mathbf{x}_j, \pi) = \Pr(yes_q|\pi)\ \gamma(j, r) + \Pr(no_q|\pi)\ (d_1(\mathbf{x}_j|j, r)\ \mu^{no}(j, r, b) + d_2(\mathbf{x}_j|j, r)),$$

$$f_2(b|\mathbf{x}_j, \pi) = \Pr(yes_q|\pi)\ (d_1(\mathbf{x}_j|j, r)\ \mu^{yes}(j, r, b) + d_2(\mathbf{x}_j|j, r)) + \Pr(no_q|\pi)\ \gamma(j, r),$$

$$\rho_j(\mathbf{x}_j) = \max_{\mathbf{x}' \text{ s.t. } \mathbf{x}'_j = \mathbf{x}_j} \sum_{j' \neq j} \mathbb{E}[u_{j'}(\mathbf{x}'_{j'})],$$

$$\gamma(j, r) = \max_{\mathbf{x}_j \notin dep(v_j^r)} \mathbb{E}[u_j(\mathbf{x}_j)] + \rho_j(\mathbf{x}_j),$$

$$\mu^{yes}(j, r, b) = \mathbb{E}^{yes_q}[v_j^r],$$

$$\mu^{no}(j, r, b) = \mathbb{E}^{no_q}[v_j^r],$$

$$d_1(\mathbf{x}_j|j, r) = \lambda_j\ C^r_{\mathbf{x}_j},$$

$$d_2(\mathbf{x}_j|j, r) = \mathbb{E}[u_j(\mathbf{x}_j)] - \lambda_j\ C^r_{\mathbf{x}_j} \mathbb{E}[v_j^r] + \rho_j(\mathbf{x}_j).$$

The value of the best LBQ on a specific LVF parameter v_j^r is $EPU(q[v_j^r, b^*])$, which can be computed using the EPU equation 4.29.

Myopic elicitation procedure with mixtures of uniforms

Using mixture-of-uniforms prior distribution over LVF parameters and local bound queries, we can implement the myopic elicitation procedure described earlier in Section 4.2.2 and test its effectiveness. The key advantage of this particular setting is that it allows us to efficiently

Input: Set of feasible outcomes $\mathbf{X}_F$, set of local bound queries Q, set of query costs $\{c_q|q \in Q\}$, sets of responses $\{\{yes_q, no_q\}|q \in Q\}$, noiseless response model $Pr(a_q|u)$, prior mixture-of-uniforms distribution $\pi = \{\langle w_k, \alpha_k, \beta_k\rangle, k = 1..K\}$ over LVF parameters v_j^k, termination criteria T
Output: Recommended outcome $\mathbf{x}^*$

$\mathbf{x}^* \leftarrow \text{argmax}_{\mathbf{x}\in\mathbf{X}_F} EU(\mathbf{x}, \pi)$
while *termination criteria T not met* **do**
  foreach *factor F_j and local value parameter v_j^r* **do**
    compute optimal bound $b^*_{v_j^r}$ according to Eq 4.43
    compute $EPU(q[v_j^r, b^*_{v_j^r}], \pi)$ according to Eq. 4.29
  end
  $q^* \leftarrow \text{argmax}_{q\in Q} EPU(q, \pi) - c_q$
  pose query q^* to the user
  if *response is "yes"* **then**
    $\pi^{yes_b} = \{\langle w_k^{yes_b}, b_k(b), \beta_k\rangle, k = 1..K\}$ (Eq. 4.34)
    $\pi \leftarrow \pi^{yes_b}$
  else if *response is "no"* **then**
    $\pi^{no_b} = \{\langle w_k^{no_b}, \alpha_k, b_k(b)\rangle, k = 1..K\}$ (Eq. 4.34)
    $\pi \leftarrow \pi^{no_b}$
  end
  $\mathbf{x}^* \leftarrow \text{argmax}_{\mathbf{x}\in\mathbf{X}_F} EU(\mathbf{x}, \pi)$
end

Figure 4.8: The myopic Bayesian elicitation procedure with LBQ queries and mixture-of-uniforms density over LVF parameters

perform continuous optimization to compute the optimal bound b for each local bound query on a specific LVF parameter (Eq. 4.43). Knowing the optimal bound for each LVF parameter, we can *enumerate* every local bound query (the number of such queries is the same as the number of GAI function parameters), compute their EVOI (Eq. 4.29), and select the best myopic query based on its expected value of information and its cost.

The detailed algorithm for the myopic elicitation is shown in Fig. 4.8. It follows the same pattern as the generic myopic elicitation algorithm discussed in Section 4.2.2 (Fig. 4.4). The decision support system computes the best local bound query to ask by 1) finding the optimal bounds $b^*_{v_j^r}$ (according to Eq. 4.43) for each LVF parameter v_j^r; 2) using those bounds to compute EPUs for local bound queries on each LVF parameter v_j^r; 3) selecting the query with the highest difference between its EPU and its cost.

Due to GAI utility decomposition into factors and closed updates of mixtures of uniforms, computing the myopically optimal local bound query at each query-response iteration is a tractable procedure. Its complexity is linear in the number of GAI parameters and exponential in the induced tree width of the factor graph, which is commonly small (the exponential complexity arises because of the need to compute the $\rho_j(\mathbf{x}_j)$ function (Eq. 4.27) for each local outcome $\mathbf{x}_j$).

In the next section, we test the effectiveness of the myopic elicitation algorithm on a 26-attribute problem with simulated users.

4.3 Experimental results

We test the myopic GAI elicitation strategy with prior density specified as a mixture of uniform distributions on a 26-attribute car-rental problem from (Boutilier et al., 2003b, 2005, 2006). The car-rental problem is modeled with 26 variables that specify various attributes of a car relevant to typical rental decisions. Variable domains range from 2 to 9 values, resulting in 61,917,364,224 possible configurations. The GAI model consists of 13 local factors, each defined on at most five variables; the model has 378 utility parameters. The full description of the car-rental domain is provided in the Appendix C.1.

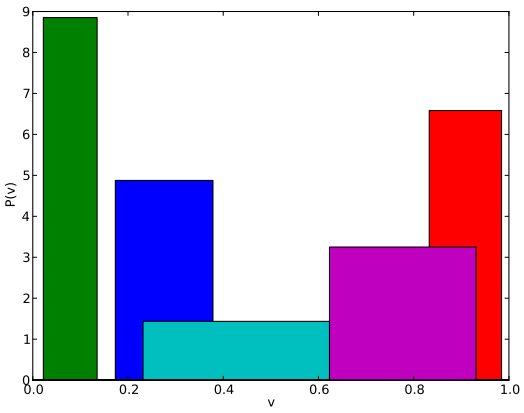
We use the myopic elicitation algorithm described in Fig. 4.8 with three types of mixture-of-uniform priors over LVF parameters:

1. a mixture of uniforms with 5 components (with the bounds chosen randomly);

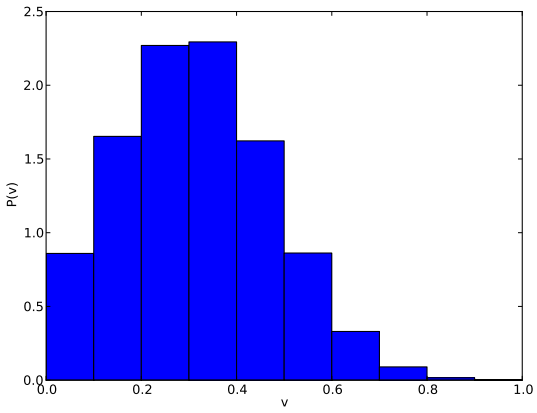
2. a mixture of 10 uniforms approximating a Gaussian distribution (with the mean chosen randomly, and the variance of 0.3);

3. a simple one-component uniform distribution.

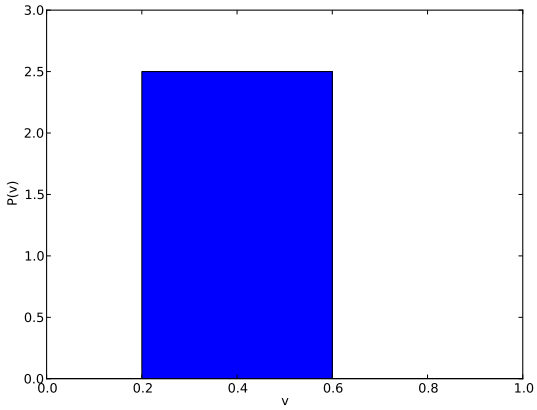
Figure 4.9 shows an example of the three types of priors.



(a) Mixture of 5 uniforms



(b) "Gaussian" mixture of 10 uniforms



(c) Uniform distribution

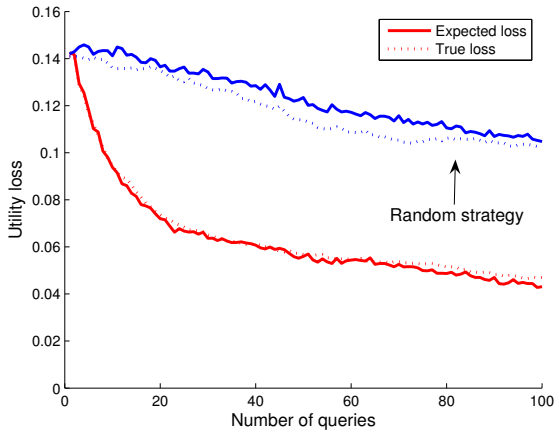
Figure 4.9: An example of the three types of mixtures of uniform distributions used as priors in our experiments. The first one (a) is a mixture of 5 uniforms; the second (b) is a mixture of 10 uniforms approximating a Gaussian distribution; and, the third one (c) is a simple one-component uniform distribution.

For all types of priors, the structure of the car-rental problem is sufficient to compute the myopically optimal query (that is, to compute the best bound for every LVF parameter and to select the query with the highest EPU) in less than one second. Therefore, this approach could support interactive real-time preference elicitation.

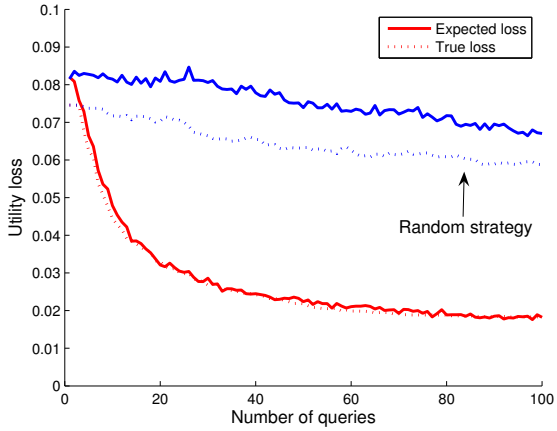
For this experiment, we assume that every query has the same cost; we stop the elicitation process after 100 queries. To provide the baseline strategy to compare against, we use the *random query strategy*. The random strategy chooses an LVF parameter (i.e., a GAI factor and one of its local configurations) to query about at random; the query bound b, however, is not selected randomly. To ensure a fair comparison between the random and Bayesian strategies, we set the bound for a random query to the expected local value of the LVF parameter, so as to give equal odds to either response.

To run the experiment, we start with the three fixed prior distributions over LVF parameters decribed above (a mixture of uniforms, a mixture of uniforms approximating a Gaussian distribution, and a simple uniform distribution), and then sample 100 "true" utility functions from each of these prior distributions. Figure 4.10 shows the averaged (over 100 simulated user utilities) performance of the Bayesian and random strategies for the three types of priors as a function of the number of elicitation queries. After each query, we compute the expected loss (see Eq. 4.17) of terminating the elicitation process by sampling 30 utility functions according to the system's belief state, and averaging the expected loss under each sampled utility function. In addition to plotting expected loss (which is one of the termination criteria available to the system), we also show true utility loss (which is not available to the system). The difference between the expected and true utility loss in most cases is small, which validates the use of expected loss as an appropriate termination criterion.

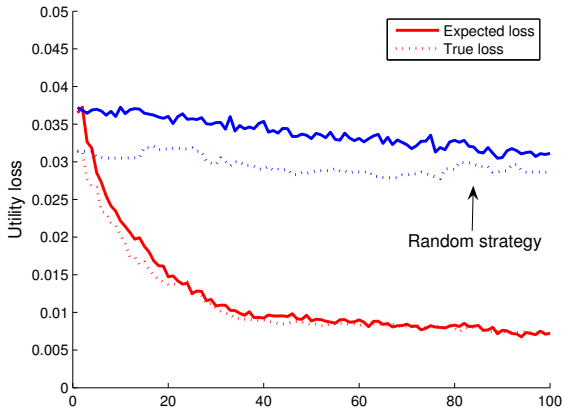
Each subfigure in Figure 4.10 compares the performance of the myopic EVOI strategy and the random strategy described above. For all three types of priors, the myopically optimal EVOI strategy is clearly superior to the random query strategy, which at best reduces the true loss by about 20% after 100 queries (for the uniform and mixture-of-uniforms prior), and at



(a) Expected and true utility loss (uniform prior)



(b) Expected and true utility loss (mixture prior)



(c) Expected and true utility loss (Gaussian-like prior)

Figure 4.10: The performance of the random and the myopic Bayesian elicitation strategies on the car-rental domain for three different types of priors. The graphs show expected (solid lines) and true (dotted lines) utility loss as a function of the number of elicitation queries. The random strategy curves are on the top, and the EVOI strategy curves are on the bottom.

worst by only 2% for the Gaussian-like prior. The EVOI strategy cuts the utility loss by 50% after 20 queries in Figure 4.10 (a), and after 16 queries in Figures 4.10 (b) and 4.10 (c). When evaluating the performance of the EVOI strategy, we should keep in mind that the problem is large (378 parameters), and the queries are limited to *local bound* queries (which do not provide as much information as, for example, direct utility queries).

Comparing different priors, we can see that because the "Gaussian" prior has less entropy than the mixture-of-uniforms or the uniform prior, its starting utility loss value (before any elicitation queries) is much lower than for the other two priors (less than 0.04 vs. 0.08 and 0.14). However, the performance of the random versus myopic EVOI strategy follows a similar pattern across the three types of priors: the random strategy achieves only a very small decrease in utility loss when compared to the myopically optimal strategy.

4.4 Conclusion

4.4.1 Related work

The origins of the Bayesian modeling of uncertainty over utilities dates back to early research in game theory and decision theory: probabilistic modeling of possible payoff functions provides the foundation to the well-established field of Bayesian games (Harsanyi, 1967, 1968); a related concept of *adaptive utility* is discussed by Cyert and de Groot (1979); de Groot (1983); and, Weber (1987) proposes using expectations over utility functions as a possible criterion for decision making with incomplete preference information. The hierarchical Bayesian techniques used in the marketing field of conjoint analysis also treat utilities as random variables, drawn from a distribution that aggregates the utilities of users from a specific population.

In the AI community, the Bayesian elicitation is used by Chajewska and Koller (2000); Chajewska et al. (2000); Boutilier (2002); Braziunas and Boutilier (2005); Guo and Sanner (2010); Bonilla, Guo, and Sanner (2010); Viappiani and Boutilier (2010). Among the probability distributions used to model uncertainty over utility functions, we find mixtures of Gaussians

(Chajewska et al., 2000), mixtures of truncated Gaussians (Boutilier, 2002), Gaussians (Guo and Sanner, 2010), Gaussian processes (Bonilla et al., 2010), mixtures of uniforms (Boutilier, 2002; Wang and Boutilier, 2003; Braziunas and Boutilier, 2005), and Beta distributions (Abbas, 2004). Such probability models are chosen to ensure that they are either closed under updates due to user responses, or easy to refit after each response. Viappiani and Boutilier (2010) examine myopic EVOI optimization of *choice queries* in settings where a user is asked to select the most preferred option from a set; they employ a logistic reponse model and consider both noiseless and noisy responses.

4.4.2 Contributions and limitations

In this chapter, we demonstrate the feasibility of the Bayesian approach to elicitation of local GAI utility parameters (all other work assumed flat utility representations). In particular, we show that if the priors over local parameters are specified using mixtures of uniforms, then the best myopic local bound query on a specific LVF parameter can be computed analytically (Braziunas and Boutilier, 2005). This is due to the fact that mixtures of uniforms are closed under updates resulting from bound queries, which makes it possible to maintain an exact density over utility parameters throughout the elicitation process. We use this result to develop a tractable procedure for myopically optimal preference elicitation. Experimental results with simulated user utilities confirm the benefits of our approach and show that it can potentially support interactive real-time elicitation.

The elicitation algorithm described in this chapter is meant to demonstrate the feasibility of Bayesian elicitation. It has not been tested with real users (however, Chapter 6 describes a user study with a recommendation system that uses the same general elicitation framework, based on a sequence of interactions (queries and responses) between the user and the system). Our approach relies on many assumptions; many aspects can be improved with further research.

The general elicitation framework, described in Section 4.2.1 (Fig. 4.3) is quite rigid: users

interact with the decision support system by providing predefined responses to a limited set of queries. We assume that users can clearly articulate their preferences and provide meaningful, accurate and consistent responses. We ignore issues of framing and potential biases that are introduced due to the order of queries (Pu et al., 2003). Nonetheless, we hope that the basic elicitation framework introduced in this chapter can serve as a basis for a practical decision support system that also addresses human-centered elicitation issues.

There are many ways to represent probabilistic uncertainty over utilities. Here, we only explore the setting in which the system has a mixture-of-uniforms prior over LVF parameters. This imposes a strong probabilistic independence assumption on the density that represents utility function uncertainty. Investigating other forms of prior distributions (over possibly different utility parameter sets) remains an open research direction. A related problem is acquisition of prior knowledge. Priors could be provided by experts, or learned from data (Chajewska et al., 1998). Although public utility databases are very scarce, their availability is increasing (Portabella Clotet and Rajman, 2006; Braziunas and Boutilier, 2010). When no prior information is available, we need a different approach to preference elicitation; one such approach is presented in the next two chapters.

Out of the four basic query types (see Section 3.3), in this chapter we concentrate on local bound queries only. Comparison queries present serious challenges when uncertainty is represented by a parametric probability model since responses to comparison queries impose "diagonal" (rather than axis-parallel) constraints on posterior distributions. This requires sampling and refitting for belief state maintenance and also complicates query optimization. However, incorporation of more types of queries is vital for practical systems, since users are likely to be more comfortable with simpler binary comparison queries than bound queries.

Finally, our proposed myopic elicitation is not sequentially optimal because it does not consider the impact of future queries when computing the value of a belief state. Solving for the sequentially optimal query policy is equivalent to solving a POMDP with a continuous state space, which is a very hard problem (Boutilier, 2002). Several approaches for solving the pref-

erence POMDP have been proposed, including value function approximation (Boutilier, 2002) and searching for good finite controller policies (Braziunas and Boutilier, 2004). However, more research is needed to scale these solutions to realistic problems. Another direction would be for the system to perform more than one-step lookahead *online* when evaluating queries (if the system has enough time between queries). With a deep lookahead horizon, the value of a query would approach its sequentially optimal value.

Chapter 5

Minimax regret based elicitation

Contents

This chapter deals with decision making and elicitation of GAI utilities when no prior probabilistic information about user preferences is available. The key ideas and results in this chapter were first published by Braziunas and Boutilier (2007).

5.1 Decisions with partial utility information

In contrast to the previous chapter, here we deal with a scenario in which we only have knowledge of a *set* of functions $\mathbf{U}$ to which the true user utility function belongs. Generally, such a set is described by a collection of bounds, or, more generally, arbitrary constraints, on the user's utility function parameters. Without probabilistic information on the distribution of possible utility functions, we advocate a solution that minimizes *maximum regret* over the space of possible utilities. Such *minimax regret* bounds the worst-case loss a user could experience given this uncertainty over utilities. In case further preference information is potentially available, a minimax regret based elicitation policy can be employed to reduce utility uncertainty to the extent where an (approximately) optimal decision can be recommended.

As in the Bayesian elicitation scenario, we are concerned with two main issues: how to make decisions when full utility information is not available, and how to select good elicitation queries when a user is willing to provide additional preference information.

5.1.1 Uncertainty representation

In this section, we describe how to represent strict uncertainty over GAI utilities by using constraints on GAI model parameters. First, we describe the relevant notation (already introduced in the previous chapters).

Notation

The outcome set $\mathbf{X} = X_1 \times \cdots \times X_N$ is defined by instantiations of N attributes $X_1, X_2, \ldots, X_N$, each with finite domains. Given an index set $I \subseteq \{1, \ldots, N\}$, $\mathbf{X}_I = \times_{i \in I} X_i$ is the set of *partial outcomes* (or *suboutcomes*) restricted to attributes indexed by I. We also assume a collection of M attribute subsets, or factors, that cover the set of all attributes: $F_1 \cup F_2 \cdots \cup F_M = \{X_1, X_2, \ldots, X_N\}$. A factor $F_I = \{X_i \mid i \in I\}$ contains the attributes whose indices are in the index set $I \subseteq \{1, \ldots, N\}$. For a factor F_j, $\mathbf{x}_{I_j}$, or simply $\mathbf{x}_j$, is a particular instantiation of its attributes. The factors (and their associated sets of indices) are enumerated from 1 to M: $F_1, F_2, \ldots, F_M$. To simplify the notation, we write $F_j = F_{I_j}$ (and $\mathbf{x}_j = \mathbf{x}_{I_j}$).

Let $\mathbf{x}^0 = (x_1^0, x_2^0, \ldots, x_N^0)$ denote a special outcome in $\mathbf{X}$ designated as the *reference outcome* (see Section 3.1.3). By using a reference outcome, any outcome $\mathbf{x} \in \mathbf{X}$ can be "projected" to a subset space indexed by I, resulting in the outcome $\mathbf{x}[I]$. For any $\mathbf{x} \in \mathbf{X}$, $\mathbf{x}[I]$ is an outcome where attributes of $\mathbf{x}$ not indexed by I are clamped at the reference values.

Each factor has $N_j = |\mathbf{X}_j|$ local (partial) outcomes, i.e., possible instantiations of factor attributes. A *basic outcome* $\mathbf{b}$ for factor F_j is any outcome $\mathbf{x}$ with attributes outside factor F_j set to the reference level: $\mathbf{b} = \mathbf{x}[I_j]$ for some $\mathbf{x} \in \mathbf{X}$. Each factor F_j has N_j basic outcomes, corresponding to N_j partial outcomes of F_j. The k^{th} basic outcome for factor F_j is denoted as $\mathbf{b}^{j,k}$.

The GAI factors are assumed to be generalized additively independent (see Definition 3.1); therefore, user preferences can be represented by a GAI utility function $u(\mathbf{x}) = \sum_{j=1}^{M} u_j(\mathbf{x}_j)$ (Theorem 3.1).

Two parametric representations of GAI utilities

In Chapter 3, we introduced two parameterized GAI model representations that rely on the notions of GAI parameters, local value parameters, and structure coefficients. *GAI parameters* (also referred to as *basic outcome parameters*) $\theta_j^k = u(\mathbf{b}^{j,k})$ represent utility values of basic outcomes. We let $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ to denote a vector of all basic outcome parameter values ($\boldsymbol{\Theta}$ is the space of all possible GAI parameter vectors):

$$\boldsymbol{\theta} = (\theta_1^1, \theta_1^2, \ldots, \theta_j^1, \theta_j^2, \ldots, \theta_j^{N_j}, \ldots, \theta_M^{N_M}).$$

Structure coefficients encode attribute decomposition into factors (see Section 3.2.2 for details). For each factor F_j, $\mathbf{C}_j$ is a square $N_j \times N_j$ matrix of integer coefficients, whose rows specify the linear combination of GAI parameters defining factor subutility values. For each factor outcome $\mathbf{x}_j$, $\mathbf{C}_{\mathbf{x}_j}$ is the $ind(\mathbf{x}_j)$ row of $\mathbf{C}_j$ ($ind(\mathbf{x}_j)$ is the index of the local configuration $\mathbf{x}_j$), and $C_{\mathbf{x}_j}^k = C_j[ind(\mathbf{x}_j), k]$ is the k^{th} entry in that row vector.

The first parameterization (Eq. 3.15, 3.22) defines the utility of any outcome $\mathbf{x} \in \mathbf{X}$ as a linear combination of GAI parameters:

$$u(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^{M} \sum_{k=1}^{N_j} C_{\mathbf{x}_j}^k \, \theta_j^k.$$

This representation is *global*, in the sense that all the parameters are semantically defined with respect to full outcomes (GAI parameters are utilities of basic outcomes, and basic outcomes are full outcomes).

The second GAI representation uses mostly local LVF parameters. An *LVF (local value function) parameter* $v_j^k = v_j(\mathbf{x}_j^k)$ denotes the local value of factor configuration $\mathbf{x}_j^k$. Such local value parameters can be assessed by local queries that consider only attributes in the relevant factor and its conditioning set (see Section 3.3). The local GAI representation also uses a small number ($2M$) of global *anchor parameters* $\theta_1^\top, \theta_1^\bot, \ldots, \theta_M^\top, \theta_M^\bot$ that are necessary for utility calibration across different factors. The global parameters $\theta_j^\top = u(\mathbf{b}^{j,\top})$ and $\theta_j^\bot = u(\mathbf{b}^{j,\bot})$ define the utilities of the best and worst basic outcomes for factor F_j. The local parameterization

of the GAI utility function (Eq. 3.20, 3.23) is as follows:

$$u(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^{M} \left[(\theta_j^{\top} - \theta_j^{\perp}) \sum_{k=1}^{N_j} C_{\mathbf{x}_j}^k \; v_j^k \right].$$

Since structure coefficients $\mathbf{C}_j$ are fixed given the GAI factor decomposition, a GAI utility function is fully determined by either the global parameters $\{\theta_1^1, \ldots, \theta_j^k, \ldots, \theta_M^{N_M}\}$ or $2M$ global parameters $\{\theta_1^{\top}, \theta_1^{\perp}, \ldots, \theta_M^{\top}, \theta_M^{\perp}\}$ and the local LVF parameters $\{v_1^1, \ldots, v_j^k, \ldots, v_M^{N_M}\}$. Global GAI parameters and local LVF parameters are linearly related *if anchor parameters are fully known*:

$$\theta_j^k = (\theta_j^{\top} - \theta_j^{\perp})\, v_j^k + \theta_j^{\perp},$$
$$v_j^k = \frac{1}{\theta_j^{\top} - \theta_j^{\perp}}\, \theta_j^k - \frac{\theta_j^{\perp}}{\theta_j^{\top} - \theta_j^{\perp}}.$$

Therefore, information about local parameters can be used to assess global parameters, and vice versa.

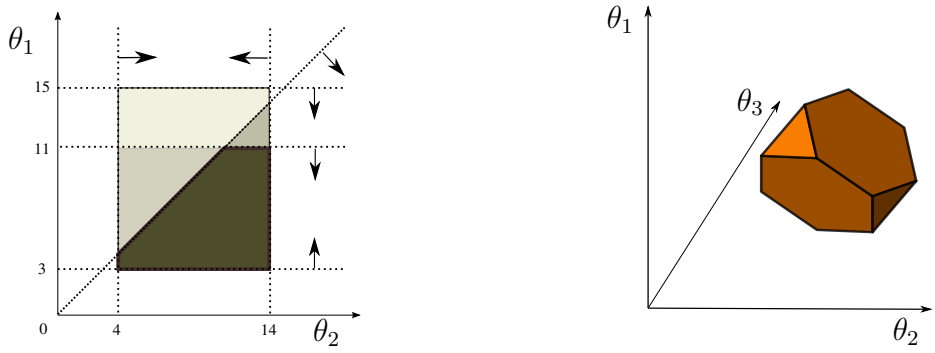
Feasible utility functions as a convex polytope in GAI parameter space

Uncertainty over utilities can be explicitly represented by set $\mathbf{U}$ of *feasible* utility functions, namely, those consistent with the system's knowledge of the user's preferences (e.g., based on responses to elicitation queries asked so far). The set $\mathbf{U}$ is updated—reduced in size—when new preference information is received during the elicitation process. In contrast to the Bayesian case, no information is available about the relative likelihood of the different utility functions in $\mathbf{U}$.

For GAI utilities, the utility space $\mathbf{U}$ can be represented by all feasible basic outcome parameter values $\boldsymbol{\Theta}_{\mathbf{U}} \subseteq \boldsymbol{\Theta}$. In this chapter, we primarily use the first parameterized representation of GAI utility functions (Eq. 3.15):

$$u(\mathbf{x}) = \sum_{j=1}^{M} \sum_{k=1}^{N_j} C_{\mathbf{x}_j}^k \; \theta_j^k.$$

The set of feasible utilities $\mathbf{U}$ is fully determined by the set of feasible parameter vectors $\boldsymbol{\Theta}_{\mathbf{U}}$. If the feasible utility space $\mathbf{U}$ is characterized by a set of *linear* constraints $\mathcal{U}$ on utility



(a) 2D polytope (b) 3D polytope

Figure 5.1: An example of two-dimensional (a) and three-dimensional (b) polytopes in the utility parameter space, defined by the set $\mathcal{U}$ of linear constraints on utility parameters. The two-dimensional polytope (a), represented by the shaded area, is defined by the sequential addition of linear *bound* and *comparison* constraints $\theta_1 \leq 15, \theta_1 \geq 3, \theta_2 \leq 14, \theta_2 \geq 4, \theta_2 \geq \theta_1, \theta_1 \leq 11$.

function parameters, then it is a *convex polytope* in the utility parameter space. Figure 5.1 shows examples of two-dimensional and three-dimensional polytopes defined by the set $\mathcal{U}$ of linear constraints on utility parameters $\boldsymbol{\Theta}$.

5.1.2 Decision criterion: minimax regret

Without probabilistic information on the distribution of possible utility functions, the *minimax regret* (MMR) decision criterion guarantees worst-case bounds on the quality of the decision made under strict uncertainty and is therefore reasonable in many real-world scenarios (Savage, 1951; Boutilier et al., 2001; Wang and Boutilier, 2003; Boutilier et al., 2004c, 2006). It prescribes an outcome that minimizes *maximum regret* with respect to all possible realizations of the user's utility function.

Let $\mathbf{X}_F \subseteq \mathbf{X}$ be the set of *feasible* outcomes. Such a set could be explicitly enumerated as a list of items in a database, or defined implicitly by a collection of hard constraints $\mathcal{H}$ on attribute instantiations. Minimax regret can be defined in stages, building on the notions of pairwise regret and maximum regret (Boutilier et al., 2001).

Definition 5.1 (Pairwise regret) The *pairwise regret* of choosing $\mathbf{x}$ instead of $\mathbf{y}$, when the

$u(\mathbf{x})$

	u_1	u_2	u_3	u_4
$\mathbf{x}^1$	2	2	0	1
$\mathbf{x}^2$	1	1	1	1
$\mathbf{x}^3$	0	4	0	0
$\mathbf{x}^4$	1	3	0	0

(a) Utilities of outcomes

$R(\mathbf{x}^i, \mathbf{x}^j, U)$

$\max_u R_u(\mathbf{x}^i, \mathbf{x}^j)$

	$\mathbf{x}^1$	$\mathbf{x}^2$	$\mathbf{x}^3$	$\mathbf{x}^4$
$\mathbf{x}^1$	0	1	2	1
$\mathbf{x}^2$	1	0	3	2
$\mathbf{x}^3$	2	1	0	1
$\mathbf{x}^4$	1	1	1	0

(b) Pairwise regrets

$MR(\mathbf{x})$

$\max_y R(\mathbf{x}, \mathbf{y}, U)$

$\mathbf{x}^1$	2
$\mathbf{x}^2$	3
$\mathbf{x}^3$	2
$\mathbf{x}^4$	1

(c) Maximum regrets

Table 5.1: A small example of a decision scenario with four feasible utilities u_1, u_2, u_3, u_4 and four feasible outcomes $\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3, \mathbf{x}^4$. The matrix (a) lists utilities of outcomes; the matrix (b) specifies pairwise max regrets of choosing some outcome $\mathbf{x}^i$ instead of $\mathbf{x}^j$; maximum regrets of choosing each outcome are listed in Table (c). The minimax regret-optimal outcome is $\mathbf{x}^4$.

user utility function is $u \in \mathbf{U}$, is simply the difference between the utilities of the two outcomes:

$$R_u(\mathbf{x}, \mathbf{y}) = u(\mathbf{y}) - u(\mathbf{x}). \tag{5.1}$$

Since the exact utility function u is often unknown, we also define the *pairwise max regret* with respect to the whole feasible utility region $\mathbf{U}$:

$$R(\mathbf{x}, \mathbf{y}, \mathbf{U}) = \max_{u \in \mathbf{U}} (u(\mathbf{y}) - u(\mathbf{x})) = \max_{u \in \mathbf{U}} R_u(\mathbf{x}, \mathbf{y}). \tag{5.2}$$

Thus, the pairwise max regret with respect to the utility region $\mathbf{U}$ is defined by choosing a utility function within that region that maximizes the pairwise regret $R_u(\mathbf{x}, \mathbf{y})$. This utility

$$u^w = \arg \max_u R_u(\mathbf{x}, \mathbf{y})$$

will be referred to as an "adversary's utility" or "witness utility", since it represents a point in the utility space $\mathbf{U}$ that maximizes the regret.

Example 5.1 Consider the decision situation summarized in Table 5.1(a). In this small example, the feasible utility set $\mathbf{U} = \{u_1, u_2, u_3, u_4\}$ is finite and consists of four different utility

functions. The feasible outcome set $\mathbf{X}_F = \{\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3, \mathbf{x}^4\}$ contains four feasible outcomes, whose utilities are specified in Table 5.1(a). In Table 5.1(b), the pairwise max regret of choosing $\mathbf{x}^2$ rather than $\mathbf{x}^3$ is 3 (with the witness utility u_2), while the regret of choosing $\mathbf{x}^3$ rather $\mathbf{x}^2$ is 1 (with witness utilities u_1, u_2, and u_4). Pairwise max regret is not symmetrical. It could be negative, if $\mathbf{x}$ is preferred to $\mathbf{y}$ under any possible utility function in $\mathbf{U}$.

Definition 5.2 (Maximum regret) The *maximum regret* of choosing $\mathbf{x}$ is

$$MR(\mathbf{x}, \mathbf{U}) = \max_{\mathbf{y} \in \mathbf{X}_F, u \in \mathbf{U}} R_u(\mathbf{x}, \mathbf{y}) = \max_{\mathbf{y} \in \mathbf{X}_F} R(\mathbf{x}, \mathbf{y}, \mathbf{U}). \tag{5.3}$$

The maximum regret value is achieved by finding both the outcome $\mathbf{y}$ and utility u that make the regret of choosing $\mathbf{x}$ as large as possible. The values $\langle \mathbf{y}^w, u^w \rangle = \arg\max_{\mathbf{y} \in \mathbf{X}_F, u \in \mathbf{U}} R_u(\mathbf{x}, \mathbf{y})$ that achieve the maximum regret $MR(\mathbf{x}, \mathbf{U})$ will be known as the "witness" or "adversary's" choice. Since $y^w = \arg\max_{\mathbf{y} \in \mathbf{X}_F} R_{u^w}(\mathbf{x}, \mathbf{y})$, y^w is the optimal outcome for u^w.

Example 5.2 In the example shown in Table 5.1(c), the maximum regret of choosing $\mathbf{x}^1$ is 2, obtained by taking the maximum pairwise regret across all columns of the matrix 5.1(b).

Definition 5.3 (Minimax regret and minimax regret-optimal solution) Finally, the outcome that minimizes max regret is the *minimax optimal outcome*:

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x} \in \mathbf{X}_F} \max_{\mathbf{y} \in \mathbf{X}_F, u \in \mathbf{U}} R_u(\mathbf{x}, \mathbf{y}) = \operatorname*{argmin}_{\mathbf{x} \in \mathbf{X}_F} MR(\mathbf{x}, \mathbf{U}). \tag{5.4}$$

The minimax regret optimal outcome $\mathbf{x}^*$, together with the adversary's choice of an outcome $\mathbf{y}^w$ and utility u^w that maximize the regret of $\mathbf{x}^*$ comprise the *solution* triple $\langle \mathbf{x}^*, \mathbf{y}^w, u^w \rangle$ of a minimax regret problem. For a feasible utility set $\mathbf{U}$, the minimax regret level $MMR(\mathbf{U})$ is determined by the solution triple:

$$MMR(\mathbf{U}) = \min_{\mathbf{x} \in \mathbf{X}_F} \max_{\mathbf{y} \in \mathbf{X}_F} \max_{u \in \mathbf{U}} [u(\mathbf{y}) - u(\mathbf{x})] = u^w(\mathbf{y}^w) - u^w(\mathbf{x}^*). \tag{5.5}$$

By choosing $\mathbf{x}^*$, the user is guaranteed to be no more than $MMR(\mathbf{U})$ away from the optimal decision utility. Furthermore, any other outcome $\mathbf{x} \neq \mathbf{x}^*$ has $MR(\mathbf{x}, \mathbf{U}) \geq MMR(\mathbf{U})$, i.e., any other outcome can result in a loss at least as great as the worst case loss of $\mathbf{x}^*$ under *some* realization $u \in \mathbf{U}$ of the user's utility function.

Example 5.3 In the example shown in Table 5.1(c), minimax regret is 1, obtained by the solution $\mathbf{x}^* = \mathbf{x}^4, \mathbf{x}^w = \mathbf{x}^1, u^w = u_1$ (there are other adversary choices that achieve the same MMR level as well).

The rest of the chapter deals with the application of the minimax decision criterion to GAI models.

5.2 Minimax regret in GAI models

In factored utility models, uncertainty over utilities can be represented by constraints over utility function parameters. In our setting, we assume the GAI utility model with linear constraints $\mathcal{U}$ on GAI parameters $\mathbf{\Theta_U}$ (see Figure 5.1). With appropriate modeling of feasible outcomes and feasible utilities, we can exploit the GAI structure in computing the minimax regret optimal choice.

In this chapter, we extend the previous work on applying minimax regret to decision making with and elicitation of GAI utility models (Boutilier et al., 2001, 2003b, 2005, 2006) in several new directions. Boutilier et al. (2006) represent uncertainty over GAI utilities by upper and lower bounds on *subutility function* parameters and show how to compute MMR using mixed integer linear programs that take advantage of this specific uncertainty representation; the bounds can be tightened using bound queries with binary responses. Based on our results from Chapter 3, we improve the previous approaches by 1) using a GAI parameterization that preserves local structure semantics and allows us to employ *semantically sound* local queries (avoiding previously ignored "calibration" issues of local utility values across different factors); 2) using *arbitrary* linear constraints to represent feasible utilities; and, 3) incorporating *all* types of queries (not just bound queries) described in Sec 3.3 into the elicitation framework.[1] In addition to previously considered decision settings where feasible outcomes are

[1]Boutilier et al. (2006) also provide mixed linear integer formulations of MMR with arbitrary linear constraints, but do not experiment with them.

defined by hard constraints on attribute values, we also deal with database problems, where feasible outcomes are *explicitly* specified as items in a multiattribute database.

The current section deals with computation of minimax regret in GAI models (while the following section addresses minimax regret-driven *elicitation* of GAI models). We first introduce two sets of constraints that define the feasible utility set $\mathbf{U}$. Then, we show how the max pairwise regret (given two specific outcomes in $\mathbf{X}_F$) can be computed by solving a linear program. The computation of pairwise regret is the same for both configuration and database problems. Computing maximum regret and minimax regret requires different techniques, which we describe separately for configuration and database problems.

5.2.1 Constraints on GAI utility parameters

Up to this point, we assumed that uncertainty over utility functions is represented by the set $\mathcal{U}$ of linear constraints that define the set of feasible GAI parameters $\mathbf{\Theta}_\mathbf{U}$. Here, we introduce an additional GAI structure constraint set $\mathcal{G}$. GAI structure constraints $\mathcal{G}$ can be thought of as a part of $\mathcal{U}$; however, since they depend only on the GAI function *structure* and remain static with additional preference information, it is useful to consider them separately.

5.2.1.1 GAI structure constraints $\mathcal{G}$

The GAI structure constraints $\mathcal{G}$ are equality constraints on certain GAI parameters θ_j^k. Because of the GAI network structure, some basic outcomes belonging to different factors are, in fact, the same. That is, the basic outcome sets B_j for different factors F_j *are not disjoint*; some outcome $\mathbf{x}$ might be in several basic outcome sets. One example is the reference outcome $\mathbf{x}^0$: by definition, $\mathbf{x}^0 = \mathbf{x}^0[I_1] = \mathbf{x}^0[I_2] \ldots = \mathbf{x}^0[I_M]$, which is a member of *all* basic outcome sets $B_1, \ldots, B_M$. Let $\theta_j^0 = u(\mathbf{x}^0[I_j])$. Then, since $\theta_j^0 = u(\mathbf{x}^0[I_j]) = u(\mathbf{x}^0)$ for all $j = 1, \ldots, M$, $\mathcal{G}$ always contains $M - 1$ constraints $\theta_1^0 = \theta_2^0 = \ldots = \theta_M^0$. In addition, if the factors are not disjoint, there are other basic outcomes that belong to several factors. In the example in Table 5.2, $\mathbf{b}^{1,1} = \mathbf{b}^{2,1} = (x^1, y^1, z^1)$, and therefore, $\theta_1^1 = \theta_2^1$, in addition to the reference

$\mathbf{x}_1$	
$\mathbf{x}_1^1$ =	$\underline{x}^1 y^1$
$\mathbf{x}_1^2$ =	$x^2 y^1$
$\mathbf{x}_1^3$ =	$\underline{x^1 y^2}$
$\mathbf{x}_1^4$ =	$x^2 \underline{y}^2$

(a) Factor F_1

$\mathbf{x}_2$	
$\mathbf{x}_2^1$ =	$y^1 \underline{z}^1$
$\mathbf{x}_2^1$ =	$\underline{y^2 z^1}$
$\mathbf{x}_2^1$ =	$y^1 z^2$
$\mathbf{x}_2^1$ =	$\underline{y}^2 z^2$

(b) Factor F_2

$\mathbf{x}$	
$\underline{x}^1 y^1 \underline{z}^1$ =	$\mathbf{b}^{1,1} = \mathbf{b}^{2,1}$
$x^2 y^1 \underline{z}^1$ =	$\mathbf{b}^{1,2}$
$\underline{x^1 y^2 z^1}$ =	$\mathbf{b}^{1,3} = \mathbf{b}^{2,2} = \mathbf{x}^0$
$x^2 \underline{y^2 z^1}$ =	$\mathbf{b}^{1,4}$
$\underline{x}^1 y^1 z^2$ =	$\mathbf{b}^{2,3}$
$x^2 y^1 z^2$	
$\underline{x}^1 y^2 z^2$ =	$\mathbf{b}^{2,4}$
$x^2 \underline{y}^2 z^2$	

(c) Basic outcomes

Table 5.2: An illustrative example of a GAI model with three binary attributes x, y, z and two factors $F_1 = \{X, Y\}$ and $F_2 = \{Y, Z\}$. The reference outcome is $\mathbf{x}^0 = \underline{x^1 y^2 z^1}$; in all configurations shown above, the reference values are underlined. Table (a) enumerates all the local configurations of factor F_1, and Table (b) enumerates all the local configurations of factor F_2. Table (c) shows all outcomes in $\mathbf{X}$, and specifies which of those outcomes are basic outcomes. For example, outcome $x^2 y^2 z^2$ is not a basic outcome for any factor, whereas outcome $x^2 \underline{y^2 z^1} = \mathbf{b}^{1,4}$ is the fourth basic outcome of factor F_1 (because $\mathbf{x}_1^4 = x^2 \underline{y}^2$ is the fourth local configuration in factor F_1, and $\underline{z}^1$ is the remaining attribute *outside* the factor, fixed at the reference level). Some outcome might also be a basic outcome for *several* different factors: $\underline{x}^1 y^1 \underline{z}^1 = \mathbf{b}^{1,1} = \mathbf{b}^{2,1}$ is a basic outcome for both factors F_1 and F_2; the same holds for the reference outcome $\underline{x^1 y^2 z^1} = \mathbf{b}^{1,3} = \mathbf{b}^{2,2}$. GAI structure constraints $\mathcal{G}$ ensure that utility parameters θ_j^k reflect such equalities between basic outcomes from different factors.

constraint $\theta_1^3 = \theta_2^2$. For any two factors F_j and F_k with shared attributes, we also have $|\mathbf{X}_{I_j \cap I_k}|$ shared basic outcomes (each corresponding to a local instantiation of the shared attributes).

Because of the interdependencies among factors, the number of *independent* GAI parameters is smaller than the number of all GAI parameters θ_j^k. The size of the GAI structure constraint set $\mathcal{G}$ is exactly the difference between the number of all GAI parameters and the number of independent GAI parameters; it depends on the GAI network structure and is exponential in the size of the largest pairwise intersection between GAI factors. In practice, the set $\mathcal{G}$ can be computed by going through all GAI parameters θ_j^k, and recording the parameters that map to the same full outcome.

5.2.1.2 GAI utility constraints $\mathcal{U}$

At any point in the elicitation process, GAI utility constraints $\mathcal{U}$ reflect the current knowledge of the utility parameters θ_j^k. The constraint set $\mathcal{U}$ changes (increases in size) as more preference information is obtained, while the set of structure constraints $\mathcal{G}$ remains static, since it depends only on the GAI model structure.

At the outset, the decision support system's knowledge of user utilities might be very minimal, in which case $\mathcal{U}$ consists of only a few very loose initial constraints. In some domains, it is reasonable to impose loose upper and lower bounds on the GAI parameters. For example, in the apartment rental domain (see Chapter 6), we assume that all apartments are valued between \$400 and \$1800. Since we use a monetary scale for utilities, the initial bounds for parameters θ_k^j are $[400, 1800]$.

Responses to preference queries described in Section 3.3 add linear constraints to the set $\mathcal{U}$. For example, a global comparison query response $\mathbf{x} \succeq \mathbf{y}$ results in the constraint

$$\sum_{j=1}^{M} \sum_{k=1}^{N_j} (C_{\mathbf{x}_j}^k - C_{\mathbf{y}_j}^k)\ \theta_j^k \geq 0,$$

while a response to a local bound query $v_j(\mathbf{x}_j^i) \geq b$ adds the constraint

$$\theta_j^i - b\theta_j^\top - (1-b)\theta_j^\perp \geq 0.$$

5.2.2 Pairwise regret

The pairwise regret equation for GAI models follows directly from the parametric representation of GAI utilities:

$$\begin{aligned} R(\mathbf{x},\mathbf{y},\mathbf{U}) &= \max_{u\in\mathbf{U}} \; u(\mathbf{y}) - u(\mathbf{x}) \\ &= \max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}_{\mathbf{U}}} \left(\sum_{j=1}^{M}\sum_{k=1}^{N_j} C^k_{\mathbf{y}_j}\; \theta^k_j - \sum_{j=1}^{M}\sum_{k=1}^{N_j} C^k_{\mathbf{x}_j}\; \theta^k_j \right) \\ &= \max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \sum_{j=1}^{M}\sum_{k=1}^{N_j} (C^k_{\mathbf{y}_j} - C^k_{\mathbf{x}_j})\; \theta^k_j, \end{aligned} \tag{5.6}$$

subject to GAI structure constraints $\mathcal{G}$ and

GAI utility constraints $\mathcal{U}$.

Given two outcomes $\mathbf{x}$ and $\mathbf{y}$, represented as attribute vectors, we can determine the corresponding entries $C^k_{\mathbf{x}_j}$ and $C^k_{\mathbf{y}_j}$ in the GAI structure matrices, for every factor index j and every local configuration index $k \in 1..N_j$ (see Section 3.2.2). Therefore, $C^k_{\mathbf{y}_j} - C^k_{\mathbf{x}_j}$ can be treated as known constant. Since $\mathcal{G}$ and $\mathcal{U}$ are linear constraints, the max pairwise regret, as defined above, can be computed by solving a linear program with the number variables equal to the number of GAI parameters, and the number of constraints equal to the number of GAI utility structure constraints $\mathcal{G}$ (which, in practice, is small) and the number of utility constraints $\mathcal{U}$ (which depends on the number of elicitation queries asked).

In practice, if the number of attributes in the domain is small (tens or hundreds), the pairwise regret can be computed very fast, in a fraction of a second (as we will see below).

5.2.3 Configuration problems

Unlike computation of pairwise regret, computation of maximum regret (of choosing some outcome $\mathbf{x}$) and minimax regret depends on the problem type.

In configuration problems, the space of feasible configurations $\mathbf{X}_F$ is implicitly defined by a set of constraints $\mathcal{H}$ specifying allowable combinations of attributes. Each constraint in $\mathcal{H}$ is

defined over a subset of attributes and induces a set of legal configurations of those attributes. The set of feasible configurations $\mathbf{X}_F$ satisfies all the constraints in $\mathcal{H}$.

We assume that constraints $\mathcal{H}$ are specified in a compact logical form. For example, in the car-rental domain (see Appendix C.1), the outcome space $\mathbf{X}$ is defined by 26 attributes relevant to consumers considering a car rental, such as automobile size and class, seating and luggage capacity, safety features, etc. Logical constraints, such as "no luxury sedans have four-cylinder engines" or "economy class cars do not have cruise control" enforce feasibility.

Finding a feasible outcome given constraint set $\mathcal{H}$ is a *constraint satisfaction problem* (CSP).[1] Our goal here is more general, since we are interested in *constrained optimization*: instead of finding any feasible outcome, we are interested in finding the best (in terms of minimax regret) feasible outcome.

In this section, we show how to effectively solve the maximum regret and minimax regret optimizations in configuration domains by taking advantage of the feasible outcome space structure (compactly defined by constraints $\mathcal{H}$) and GAI model structure.

5.2.3.1 Outcome encoding using attribute and factor indicators

Before we describe how to compute maximum regret in configuration domains, we introduce a way to encode outcome and factor instantiations in GAI models using binary indicator variables. Such an encoding will be useful in formulating regret computations as *mixed integer programs* (MIPs).

Let A_i^t be an indicator variable that specifies whether attribute X_i is set to the t^{th} value:

$$A_i^t = \begin{cases} 1, \text{ if } X_i = x_i^t, \\ 0, \text{ otherwise.} \end{cases} \tag{5.7}$$

For each attribute X_i, we have $|X_i|$ indicator variables $A_i^1, \ldots, A_i^{|X_i|}$, one for each discrete value of the attribute. If the attribute takes the t^{th} value, the indicator variable A_i^t is set to one,

[1] Dechter (2003) provides a detailed overview of CSP solution methods.

while all other attribute indicator variables are set to zero. Valid outcomes in $\mathbf{X}$ can be encoded by a set $\{A_i^t\}$ of binary attribute indicators if they respect the following constraints:

$$\begin{aligned} &A_i^t \in \{0,1\}, \text{ for all } i \text{ and } t, \\ &\sum_k A_i^t = 1 \text{ for all } i \text{ (attributes can take only one value at a time).} \end{aligned} \tag{5.8}$$

In GAI models, there is also a correspondence between a full outcome instantiation $\mathbf{x}$ and consistent factor instantiations $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M$. That is, given $\mathbf{x}$, we can determine $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M$, and vice versa:

$$\mathbf{x} \longleftrightarrow (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M).$$

Just like for attribute values, we can encode factor outcome instantiations using indicator variables Z_j^k, with k ranging through all N_j possible factor instantiations:

$$Z_j^k = \begin{cases} 1, \text{ if } \mathbf{X}_j = \mathbf{x}_j^k, \\ 0, \text{ otherwise.} \end{cases} \tag{5.9}$$

Let $\mathbf{x}_j^k \ni x_i^t$ denote an instantiation relationship between a factor outcome $\mathbf{x}_j^k$ and an attribute value x_i^t. That is, $\mathbf{x}_j^k \ni x_i^t$ means that the factor F_j contains the attribute X_i, and in $\mathbf{x}_j^k$ that attribute takes value x_i^t. Factor indicators Z_j^k obey the following constraints:

$$\begin{aligned} &Z_j^k \in \{0,1\}, \text{ for all } j \text{ and } k \in 1..N_j, \\ &\sum_k Z_j^k = 1 \text{ for all } j \text{ (there can be only one factor instantiation),} \\ &\sum_{k \text{ s.t. } \mathbf{x}_j^k \ni x_i^t} Z_j^k \leq A_i^t \text{ for all } j \in 1..M, \text{ for all } i \text{ such that } X_i \in F_j, \text{ and all values } t \text{ of } X_i. \end{aligned} \tag{5.10}$$

The first two constraints are similar to those for attribute indicators. The third one ties together attribute and factor indicator constraints to ensure consistent instantiations. In particular, we need to ensure that whenever some factor is instantiated to the factor outcome $\mathbf{x}_j^k$, all attributes

in that factor take appropriate values:

$$\mathbf{x}_j^k \implies x_i^t, \text{ if } \mathbf{x}_j^k \ni x_i^t$$

$$\iff$$

$$Z_j^k \implies A_i^t, \text{ if } \mathbf{x}_j^k \ni x_i^t$$

$$\iff$$

$$Z_j^k \leq A_i^t, \text{ if } \mathbf{x}_j^k \ni x_i^t$$

Because only one factor indicator can be non-zero, we can combine combine all the relevant factor indicators into one constraint for each attribute indicator A_i^t (Eq. 5.10):

$$\sum_{k \text{ s.t. } \mathbf{x}_j^k \ni x_i^t} Z_j^k \leq A_i^t.$$

The set of attribute and indicator constraints in Eq. 5.8 and Eq. 5.10 will be denoted by $\mathcal{A}$.

5.2.3.2 Hard constraints $\mathcal{H}$

The constraint set $\mathcal{H}$ compactly specifies the set of feasible outcomes $\mathbf{X}_F \subseteq \mathbf{X}$ by restricting the choices of indicators A_i^t to allow only configurations in $\mathbf{X}_F$. One natural way to define feasible configurations is by using logical constraints, which, in turn, can be encoded as linear constraints on attribute indicators A_i^t (Chandru and Hooker, 1999). For example, in the car rental domain (see Appendix C.1), we have logical constraints, such as "economy class cars do not have cruise control". Let $A_{CarClass}^{\texttt{economy}}$ be a binary attribute indicator that specifies whether the $CarClass$ attribute is set to value `economy`; similarly, $A_{CruiseControl}^{\texttt{true}}$ indicates whether the car has cruise control. A logical constraint "economy class cars do not have cruise control" can be written as a linear constraint:

$$A_{CarClass}^{\texttt{economy}} + A_{CruiseControl}^{\texttt{true}} \leq 1.$$

5.2.3.3 Maximum regret

In GAI models, the maximum regret of choosing a particular outcome $\mathbf{x}$ is

$$\begin{aligned} MR(\mathbf{x}, \mathbf{U}) &= \max_{\mathbf{y} \in \mathbf{X}_F,\, u \in \mathbf{U}} (u(\mathbf{y}) - u(\mathbf{x})) \\ &= \max_{\mathbf{y} \in \mathbf{X}_F,\, \boldsymbol{\theta} \in \boldsymbol{\Theta}} \sum_{j=1}^{M} \sum_{k=1}^{N_j} (C^k_{\mathbf{y}_j} - C^k_{\mathbf{x}_j})\, \theta^k_j, \end{aligned} \tag{5.11}$$

subject to GAI structure constraints $\mathcal{G}$ and utility constraints $\mathcal{U}$.

We can rewrite the maximum regret optimization using indicator variables. Instead of maximizing over all $\mathbf{y}$ in $\mathbf{X}_F$, we can now maximize over a set of binary indicators Z^k_j; linear feasibility constraints $\mathcal{H}$ on attribute values will in turn restrict the choices of indicators A^t_i to allow only configurations in $\mathbf{X}_F$. The new optimization is:

$$\begin{aligned} MR(\mathbf{x}, \mathbf{U}) &= \max_{\mathbf{y} \in \mathbf{X}_F,\, \boldsymbol{\theta} \in \boldsymbol{\Theta}_\mathbf{U}} \sum_{j=1}^{M} \sum_{k=1}^{N_j} (C^k_{\mathbf{y}_j} - C^k_{\mathbf{x}_j})\, \theta^k_j, \\ &= \max_{\{A^t_i, Z^r_j\}, \boldsymbol{\theta}} \sum_{j=1}^{M} f_j(Z^r_j), \text{ subject to constraints } \mathcal{A}, \mathcal{H}, \mathcal{G}, \text{ and } \mathcal{U}, \end{aligned}$$

where

$$f_j(Z^r_j) = \begin{cases} \sum_{k=1}^{N_j} (\mathbf{C}_j[r,k] - C^k_{\mathbf{x}_j})\, \theta^k_j, \text{ if } Z^r_j = 1, \\ 0, \text{ if } Z^r_j = 0. \end{cases}$$

For effective computation, we would like to reformulate the above optimization as a linear integer program. For that purpose, we introduce M variables Y_j representing $f_j(Z^r_j)$, and encode the function $f_j(Z^r_j)$ as a series of linear constraints by using the "big-M" transformation:

$$\begin{aligned} MR(\mathbf{x}, \mathbf{U}) = \max_{\{A^t_i, Z^r_j\}, \boldsymbol{\theta}} & \sum_j Y_j \\ & \text{subject to} \\ & Y_j \le \sum_{k=1}^{N_j} (\mathbf{C}_j[r,k] - C^k_{\mathbf{x}_j})\, \theta^k_j + M_j(1 - Z^r_j) \;\; \forall j \in 1..M,\; r \in 1..N_j, \\ & \text{and constraints } \mathcal{A}, \mathcal{H}, \mathcal{G}, \text{ and } \mathcal{U}. \end{aligned} \tag{5.12}$$

Here, M_j is a large constant that provides an upper bound on $f_j(Z_j^r)$ for any Z_j^r. In the formulation above, the first constraint ensures that whenever $Z_j^r = 1$ (i.e., whenever an "adversary" picks the r^{th} local configuration in factor F_j), the value of Y_j gets "pushed up" to $\sum_{k=1}^{N_j}(\mathbf{C}_j[r,k] - C^k_{\mathbf{x}_j})\,\theta_j^k$ (since $M_j(1 - Z_j^r)$ is zero). If $Z_j^r = 0$, the corresponding constraint should not be active; in this case, the value of Y_j is limited only by the large constant M_j.

Since the objective and all constraints are now linear, the problem is a linear MIP.

Example 5.4 We refer back to Example 3.10, which used a GAI utility function with two factors $I_1 = \{1, 2\}$ and $I_2 = \{2, 3\}$, and all binary attributes. From the dependency structure, we know that

$$u_2(x_2, x_3) = u(\mathbf{x}[I_2]) - u(\mathbf{x}([I_1 \cap I_2]) = u(x_1^0, x_2, x_3) - u(x_1^0, x_2, x_3^0),$$

where x_i is a generic binary attribute that could take the values x_i^0 or x_i^1. We concentrate on the second factor, in which:

$$u_2(\mathbf{x}_2^1) = u_2(x_2^0, x_3^0) = u(x_1^0, x_2^0, x_3^0) - u(x_1^0, x_2^0, x_3^0) = 0\,\theta_2^1 + 0\,\theta_2^2 + 0\,\theta_2^3 + 0\,\theta_2^4,$$
$$u_2(\mathbf{x}_2^2) = u_2(x_2^1, x_3^0) = u(x_1^0, x_2^1, x_3^0) - u(x_1^0, x_2^1, x_3^0) = 0\,\theta_2^1 + 0\,\theta_2^2 + 0\,\theta_2^3 + 0\,\theta_2^4,$$
$$u_2(\mathbf{x}_2^3) = u_2(x_2^0, x_3^1) = u(x_1^0, x_2^0, x_3^1) - u(x_1^0, x_2^0, x_3^0) = -1\,\theta_2^1 + 0\,\theta_2^2 + 1\,\theta_2^3 + 0\,\theta_2^4,$$
$$u_2(\mathbf{x}_2^4) = u_2(x_2^1, x_3^1) = u(x_1^0, x_2^1, x_3^1) - u(x_1^0, x_2^1, x_3^0) = 0\,\theta_2^1 - 1\,\theta_2^2 + 0\,\theta_2^3 + 1\,\theta_2^4.$$

The coefficients $\mathbf{C}_{\mathbf{x}_2}$ specify which basic outcomes are involved in defining the subutility values for factor F_2:

$$\mathbf{C}_{\mathbf{x}_2^1} = \mathbf{C}[1,:] = [0,0,0,0],$$
$$\mathbf{C}_{\mathbf{x}_2^2} = \mathbf{C}[2,:] = [0,0,0,0],$$
$$\mathbf{C}_{\mathbf{x}_2^3} = \mathbf{C}[3,:] = [-1,0,1,0],$$
$$\mathbf{C}_{\mathbf{x}_2^4} = \mathbf{C}[4,:] = [0,-1,0,1].$$

Our goal is to find the maximum regret of choosing some outcome $\mathbf{x}$, say, (x_1^1, x_2^0, x_3^1). Since $\mathbf{x}_2^3 = (x_2^0, x_3^1)$, the second factor structure coefficients are $\mathbf{C}_{\mathbf{x}_2^3} = [-1, 0, 1, 0]$. The MIP for the maximum regret is as follows, with explicit constraints shown only for factor F_2:

$$MR(\mathbf{x}, \mathbf{U}) = \max_{Y_1, Y_2, \{A_1^t, A_2^t, A_3^t, t=0..1\}, \{Z_1^k, Z_2^k, \theta_1^k, \theta_2^k, k=1..4\}} Y_1 + Y_2$$

$$\text{subject to} \begin{cases} Y_1 \leq \sum_{k=1}^{4} (\mathbf{C}_1[r,k] - C^k_{\mathbf{x}_1^1})\, \theta_1^k + M_1(1 - Z_1^r), \;\; r = 1..4, \\ Y_2 \leq 0\,\theta_2^1 + 0\,\theta_2^2 + 0\,\theta_2^3 + 0\,\theta_2^4 + 100(1 - Z_2^1), \\ Y_2 \leq 0\,\theta_2^1 + 0\,\theta_2^2 + 0\,\theta_2^3 + 0\,\theta_2^4 + 100(1 - Z_2^2), \\ Y_2 \leq -1\,\theta_2^1 + 0\,\theta_2^2 + 1\,\theta_2^3 + 0\,\theta_2^4 + 100(1 - Z_2^3), \\ Y_2 \leq 0\,\theta_2^1 - 1\,\theta_2^2 + 0\,\theta_2^3 - 1\,\theta_2^4 + 100(1 - Z_2^4), \\ \mathcal{A}, \mathcal{H}, \mathcal{G}, \text{ and } \mathcal{U}. \end{cases}$$

Here, the large M-constant M_2 is 100.

5.2.3.4 Minimax regret

For a feasible utility set $\mathbf{U}$, the minimax regret level $MMR(\mathbf{U})$ can be determined by the following optimization:

$$\begin{aligned} MMR(\mathbf{U}) &= \min_{\mathbf{x} \in \mathbf{X}_F} \max_{\mathbf{y} \in \mathbf{X}_F, u \in \mathbf{U}} R_u(\mathbf{x}, \mathbf{y}) \\ &= \min_{\mathbf{x} \in \mathbf{X}_F, m} m, \\ &\qquad \text{subject to } m \geq R_u(\mathbf{x}, \mathbf{y}), \;\; \forall\, \mathbf{y} \in \mathbf{X}_F,\; u \in \mathbf{U}. \end{aligned}$$

In this way, we convert a min-max problem to a minimization problem over $\mathbf{x} \in \mathbf{X}_F$ only, at the expense of introducing an infinite (continuous) set of constraints (one constraint for each pair $\mathbf{y} \in \mathbf{X}_F$, $u \in \mathbf{U}$).

Due to the form of the constraints, we are interested in the pairwise regret $R_u(\mathbf{x}, \mathbf{y})$ as a function of $\mathbf{x}$, assuming the adversary's choice of $\mathbf{y}$ and u is known. For GAI utilities, the

utility function u is fully determined by the GAI parameter vector $\boldsymbol{\theta} = \theta_1^1 \dots \theta_1^{N_1} \dots \theta_M^{N_M}$. We denote the pairwise regret of choosing $\mathbf{x}$ given $\mathbf{y}$ and $\boldsymbol{\theta}$ as $R(\mathbf{x}|\mathbf{y}, \boldsymbol{\theta})$:

$$\begin{aligned} R(\mathbf{x}|\mathbf{y}, \boldsymbol{\theta}) = R_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) &= \sum_{j=1}^{M} \sum_{k=1}^{N_j} (C^k_{\mathbf{y}_j} - C^k_{\mathbf{x}_j})\ \theta^k_j, \\ &= \sum_{j=1}^{M} R_j(\mathbf{x}_j|\mathbf{y}_j, \boldsymbol{\theta}_j), \end{aligned} \tag{5.13}$$

where $R_j(\mathbf{x}^r_j|\mathbf{y}_j, \boldsymbol{\theta}_j) = g^r_j(\mathbf{y}, \boldsymbol{\theta})$ is the *factor regret* of choosing the r^{th} local configuration in factor F_j when adversary's choice is $\mathbf{y}$ under the utility function represented by $\boldsymbol{\theta}$:

$$R_j(\mathbf{x}^r_j|\mathbf{y}_j, \boldsymbol{\theta}_j) = \sum_{k=1}^{N_j} (C^k_{\mathbf{y}_j} - C^k_{\mathbf{x}^r_j})\ \theta^k_j = g^r_j(\mathbf{y}, \boldsymbol{\theta}).$$

Using factor indicators Z^k_j, the pairwise regret can now be written as:

$$R(\mathbf{x}|\mathbf{y}, \boldsymbol{\theta}) = \sum_{j=1}^{M} \sum_{k=1}^{N_j} g^k_j(\mathbf{y}, \boldsymbol{\theta}) Z^k_j. \tag{5.14}$$

where indicators are instantiated to encode the outcome $\mathbf{x}$.

Knowing $\mathbf{y}$ and $\boldsymbol{\theta}$, factor regrets $g^k_j(\mathbf{y}, \boldsymbol{\theta})$ can be precomputed for every factor configuration. Let $\mathbf{y}^*_{\boldsymbol{\theta}}$ be the best outcome for a utility function with parameters $\boldsymbol{\theta}$:

$$\mathbf{y}^*_{\boldsymbol{\theta}} = \operatorname*{argmax}_{\mathbf{y} \in \mathbf{X}_F} u(\mathbf{y}|\boldsymbol{\theta}).$$

By using factor indicators, we can recast the MMR computation as a linear MIP:

$$MMR(\mathbf{U}) = \min_{\{A^t_i\}, \{Z^k_j\}, m} m, \text{ subject to} \tag{5.15}$$
$$\begin{cases} m \geq \sum_{j=1}^{M} \sum_{k=1}^{N_j} g^k_j(\mathbf{y}^*_{\boldsymbol{\theta}}, \boldsymbol{\theta})\ Z^k_j, \ \ \forall\, \boldsymbol{\theta} \in \boldsymbol{\Theta}_V, \\ \mathcal{A}, \mathcal{H}, \end{cases}$$

where $\boldsymbol{\Theta}_V \subset \boldsymbol{\Theta}_{\mathbf{U}}$ is the (finite) set of vertices of the GAI utility polytope $\boldsymbol{\Theta}_{\mathbf{U}}$ defined by the constraints $\mathcal{U}$ and $\mathcal{G}$. We can restrict the space of parameters $\boldsymbol{\theta}$ because $g^r_j(\mathbf{y}, \boldsymbol{\theta}) = \sum_{k=1}^{N_j} (C^k_{\mathbf{y}_j} - C^k_{\mathbf{x}^r_j})\ \theta^k_j$ is a linear function of θ^k_j, and therefore any active constraints in Eq. 5.15 feature only some $\boldsymbol{\theta} \in \boldsymbol{\Theta}_V$ (i.e., constraints for $\boldsymbol{\theta} \in \boldsymbol{\Theta}_{\mathbf{U}} \backslash \boldsymbol{\Theta}_V$ are redundant). This makes the number

of constraints $|\mathbf{\Theta}_V|$ exponential with respect to the number of utility parameters (rather than infinite). For any realistic problem size, a direct optimization approach is not tractable.

However, the number of active constraints at the optimal solution is usually very small. We can avoid the exponential number of constraints (one for each adversarial utility choice $\boldsymbol{\theta} \in \mathbf{\Theta}_V$) by using a *constraint generation* procedure that iteratively generates a small set of active constraints at the optimal solution. Such a way of dealing with a large number of constraints is common in operations research, and can be viewed as a form of Benders' decomposition (Benders, 1962; Nemhauser and Wolsey, 1988; Boutilier et al., 2006). To apply constraint generation technique, we solve the MIP in Eq. 5.15 with only a subset of constraints, generate and add the maximally violated constraint at the solution of this relaxed MIP (by solving the max regret MIP in Eq. 5.12 for adversary's choice $\mathbf{y}, \boldsymbol{\theta}$), and repeat until no violated constraints are found.

Intuitively, at each iteration, we solve the MIP in Eq. 5.15 using only a limited set W of adversarial choices of pairs of outcomes and utilities:

$$MMR(\mathbf{U}, W) = \min_{\{A_i^t\},\{Z_j^k\},m} m, \text{ subject to} \quad (5.16)$$
$$\begin{cases} m \geq \sum_{j=1}^{M} \sum_{k=1}^{N_j} g_j^k(\mathbf{y}, \boldsymbol{\theta})\, Z_j^k, \;\; \langle \mathbf{y}, \boldsymbol{\theta} \rangle \in W \\ \mathcal{A}, \mathcal{H}. \end{cases}$$

The set W contains pairs of outcomes and utilities that the adversary can use to increase the maximum regret. In this way, we are computing minimax regret against a *restricted* adversary that can only use choices in W to compute the maximum regret of the optimal solution $\mathbf{x}$ obtained by solving the restricted MMR optimization problem in Eq. 5.16. The minimax regret value $MMR(\mathbf{U}, W)$ provides a *lower bound* on the true minimax regret $MMR(\mathbf{U})$ (which is computed by considering a "stronger" unrestricted adversary).

Given a solution $\mathbf{x}$ to the restricted MMR optimization problem, we can find the maximum regret $MR(\mathbf{x}, \mathbf{U})$ of $\mathbf{x}$ using Eq. 5.12, as well as the adversary choice $\langle \mathbf{y}^w, \boldsymbol{\theta}^w \rangle = \arg MR(\mathbf{x}, \mathbf{U})$ that achieves that regret (we introduce the $\arg$ notation to simplify the de-

Input: Set of feasible outcomes $\mathbf{X}_F$ (defined by constraints $\mathcal{H}$), set of feasible utilities $\mathbf{U}$, set of attribute and factor consistency constraints $\mathcal{A}$
Output: Minimax regret solution triple $\langle \mathbf{x}^*, \mathbf{y}^w, \boldsymbol{\theta}^w \rangle$

$LB \leftarrow -\infty$ // lower bound on minimax regret
$UB \leftarrow \infty$ // upper bound on minimax regret
$i \leftarrow 0$
$\mathbf{x}^0 \leftarrow$ arbitrary outcome in $\mathbf{X}_F$
$W^0 \leftarrow \emptyset$ // initial set of restricted adversary choices
while $LB \neq UB$ **do**
  $UB \leftarrow MR(\mathbf{x}^i, \mathbf{U})$ // using Eq. 5.12
  $\mathbf{y}^i, \boldsymbol{\theta}^i \leftarrow \arg MR(\mathbf{x}^i, \mathbf{U})$ // $\mathbf{y}^i, \boldsymbol{\theta}^i$ are adversary choices that achieve $MR(\mathbf{x}^i, \mathbf{U})$

  $i \leftarrow i + 1$
  $W^i \leftarrow W^{i-1} \cup \{\langle \mathbf{y}^{i-1}, \boldsymbol{\theta}^{i-1} \rangle\}$
  $LB \leftarrow MMR(\mathbf{U}, W^i)$ // using Eq. 5.16
  $\mathbf{x}^i \leftarrow \arg MMR(\mathbf{U}, W^i)$ // $\mathbf{x}^i$ is the outcome that achieves LB
end
$\mathbf{x}^* \leftarrow \mathbf{x}^i$
$\mathbf{y}^w, \boldsymbol{\theta}^w \leftarrow \arg MR(\mathbf{x}^*, \mathbf{U})$

Figure 5.2: MMR computation procedure using constraint generation. At each iteration, we compute the minimax-regret optimal choice $\mathbf{x}_i$ against a restricted adversary that can only use outcome and utility pairs from the set W^i.

tailed description of the algorithm below). The maximum regret $MR(\mathbf{x}, \mathbf{U})$ provides an *upper bound* on the true minimax regret, since we can always stop the process and recommend outcome $\mathbf{x}$. We add the new adversary choice $\langle \mathbf{y}^w, \boldsymbol{\theta}^w \rangle$ to the set W (thus making the adversary "stronger"), and repeat the process until the lower bound equals to the upper bound (i.e., $MMR(\mathbf{U}, W) = MR(\mathbf{x}, \mathbf{U})$ for some $\mathbf{x}$). The procedure is guaranteed to arrive at the optimal solution in a finite number of iterations; in theory, it is not guaranteed to finish until it has a full set of adversary choices W (of size $|\mathbf{X}_F|$, corresponding to every outcome in $\mathbf{X}_F$). In practice, however, the number of iterations is usually small, and, therefore, we can compute the minimax regret by solving a series of small MIPs.

The detailed algorithm is shown in Figure 5.2.

APARTMENTS

$1300	2 bedrooms apartment in Toronto Central	Furnished No dishwasher	Laundry available No storage room	Parking not available Not air-conditioned
$800	1 bedroom house in Toronto West	Furnished No dishwasher	Laundry not available No storage room	Parking not available Not air-conditioned
$650	1 bedroom basement in Toronto East	Unfurnished No dishwasher	Laundry available Storage room	Parking not available Air-conditioned
$700	1 bedroom apartment in Scarborough	Unfurnished No dishwasher	Laundry available No storage room	Parking available Not air-conditioned
$900	1 bedroom apartment in Toronto Central	Unfurnished Dishwasher	Laundry available Storage room	Parking available Air-conditioned
$860	1 bedroom house in Toronto Central	Unfurnished No dishwasher	Laundry not available Storage room	Parking not available Not air-conditioned
$1450	3 bedrooms house in Toronto Central	Unfurnished No dishwasher	Laundry not available Storage room	Parking not available Not air-conditioned
$1300	2 bedrooms house in Toronto Central	Unfurnished No dishwasher	Laundry not available Storage room	Parking not available Air-conditioned
$1600	1 bedroom apartment in Toronto Central	Furnished Dishwasher	Laundry available No storage room	Parking not available Air-conditioned
$1100	1 bedroom apartment in Toronto Central	Unfurnished No dishwasher	Laundry not available No storage room	Parking available Not air-conditioned

Figure 5.3: A screenshot of an apartment database used in the UTPREF recommendation system. Each apartment is described by nine discrete attributes and price.

5.2.4 Database problems

The MIP formulations above assume that the space of feasible configurations is defined by a set of constraints $\mathcal{H}$ specifying allowable combinations of attributes. Alternatively, the set of choices may be the elements of a *multiattribute item database*, in which the set of feasible outcomes is specified explicitly, namely, as the set of all items in the database. Instead of constraints specifying which configurations are not valid, in database problems, we have a finite (but possibly large) list of valid multiattribute outcomes. Common examples of multiattribute item databases include online store catalogs, where each product in a category can be adequately described by a set of attributes (e.g., computers or video cameras), real estate domains (houses for rent or sale), new and used car sales, and others. Figure 5.3 shows an example of an apartment rental database, where each apartment is defined by ten attributes, including price. Preference-based search of such multiattribute item databases can be effected using minimax regret as well; the computation of minimax regret is, however, different.

In a database setting, let $\mathbf{X}_F \subseteq \mathbf{X}$ be a finite set of outcomes in a database, and $D = |\mathbf{X}_F|$ be the database size. The items in the database will often be enumerated from 1 to D:

$$\mathbf{X}_F = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^D\}.$$

In principle, the minimax regret optimal solution can be determined by finding the pairwise regret for every pair of items in the database, and choosing an item with the minimum max regret. Minimax regret for a choosing an item from a database can be computed by using only the pairwise regret equation Eq. 5.6; because of the finite size of the database, we do not use equations 5.12 and 5.15 for computing max and minimax regret. For any two items $\mathbf{x}^i$ and $\mathbf{x}^j$ in the database, the pairwise regret $R(\mathbf{x}^i, \mathbf{x}^j, \mathbf{U})$ can be found using Eq. 5.6. The max regret $MR(\mathbf{x}^i, \mathbf{U})$ of choosing the i^{th} item is determined by considering its pairwise regret with each other item in $\mathbf{X}_F$. To determine the optimal item or product (i.e., that with the minimax regret), we can compute the max regret of each item in the database, and choose the one with the least

		MAX			MR
	3	2	2	7	7
MIN	3	4	6	4	6
	2	5	8	7	8
	2	8	9	3	9

Figure 5.4: An example of a pairwise regret matrix L for a four-item database. The MIN player chooses rows, while the MAX player chooses columns. The maximum regrets of each row are shown in the right column. The MIN player's choice is Row 2, with the smallest maximum regret of 6.

max regret:

$$\mathbf{x}^* = \mathbf{x}^{i^*} = \operatorname*{argmin}_{i \leq D} \max_{j \leq D} R(\mathbf{x}^i, \mathbf{x}^j, \mathbf{U}). \tag{5.17}$$

5.2.4.1 Pairwise regret matrix

Let L be the $D \times D$ matrix whose entries contain the pairwise regrets of all possible choices by the two players:

$$L[i, j] = R(\mathbf{x}^i, \mathbf{x}^j, \mathbf{U}).$$

The process of finding the MMR-optimal option can be viewed as a game between two players, in which the MIN player is trying to minimize the maximum regret, while the MAX player's (or adversary's) goal is to maximize the regret of the MIN player's choice. The MIN player plays the MMR game by picking rows, while the MAX player chooses columns. The worst-case complexity of the algorithm is $O(D^2)$ evaluations of pairwise regret. Pairwise regret computation is polynomial in the size of the GAI utility function (because each pairwise regret evaluation requires solving a linear program with a number of variables proportional to the number of GAI parameters). Figure 5.4 shows an example of the MMR matrix L, and the optimal solution.

5.2.4.2 Minimax search with pruning

The matrix game between the MIN and MAX players can also be represented by the two-ply *minimax tree* with the MIN root node n^{MIN}, D MAX nodes $\{n_i^{MAX}, i = 1, \ldots, D\}$, and D^2

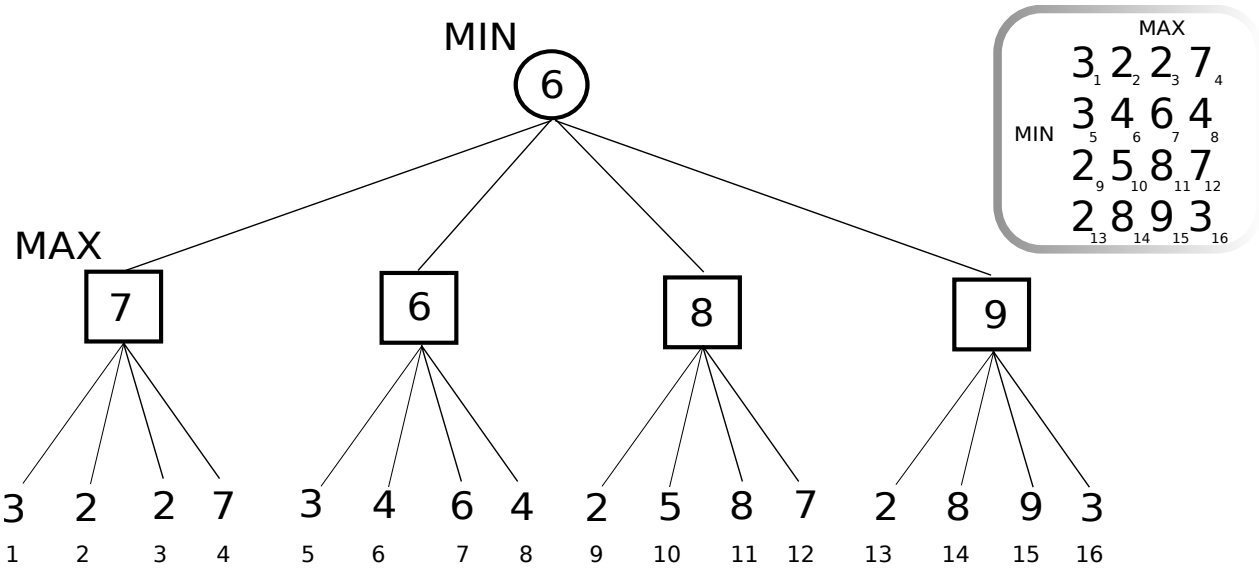


Figure 5.5: The minimax search tree for the four-item regret matrix introduced in Fig. 5.4. The small numbers indicate the order of search steps (in this case, it's the depth-first traversal).

terminal (leaf) nodes $\{n_{ij}^T, i = 1, \ldots, D\ , j = 1, \ldots, D\}$. Each MIN and MAX node has D directed edges, corresponding to D choices (i.e., outcomes in $\mathbf{X}_F$) available to each player. The unique path to each terminal node represents a specific pair of choices by the MIN and MAX players. The value of a terminal node n_{ij}^T reachable by following edge i from the root MIN node and then edge j from the MAX node is the pairwise regret of $R(\mathbf{x}^i, \mathbf{x}^j, \mathbf{U})$ of choosing $\mathbf{x}^i$ rather than $\mathbf{x}^j$. The terminal values can be "backed up" to provide values for the internal MAX nodes and the root MIN node. The value of a terminal node n_{ij}^T is

$$v(n_{ij}^T) = R(\mathbf{x}^i, \mathbf{x}^j, \mathbf{U}).$$

The value of a MAX node n_i^{MAX} is the maximum of the values of its children (terminal nodes):

$$v(n_i^{MAX}) = \max_{j \in children(n_i^{MAX})} v(n_{ij}^T).$$

The value of the root MIN node n^{MIN} is the minimum of the values of its children (MAX nodes):

$$v(n^{MIN}) = \min_{i \in children(n^{MIN})} v(n_i^{MAX}).$$

To compute the MMR optimal choice, we find the values of MAX nodes, and then choose the MAX node with the smallest value. Figure 5.5 shows a minimax search tree for the sample

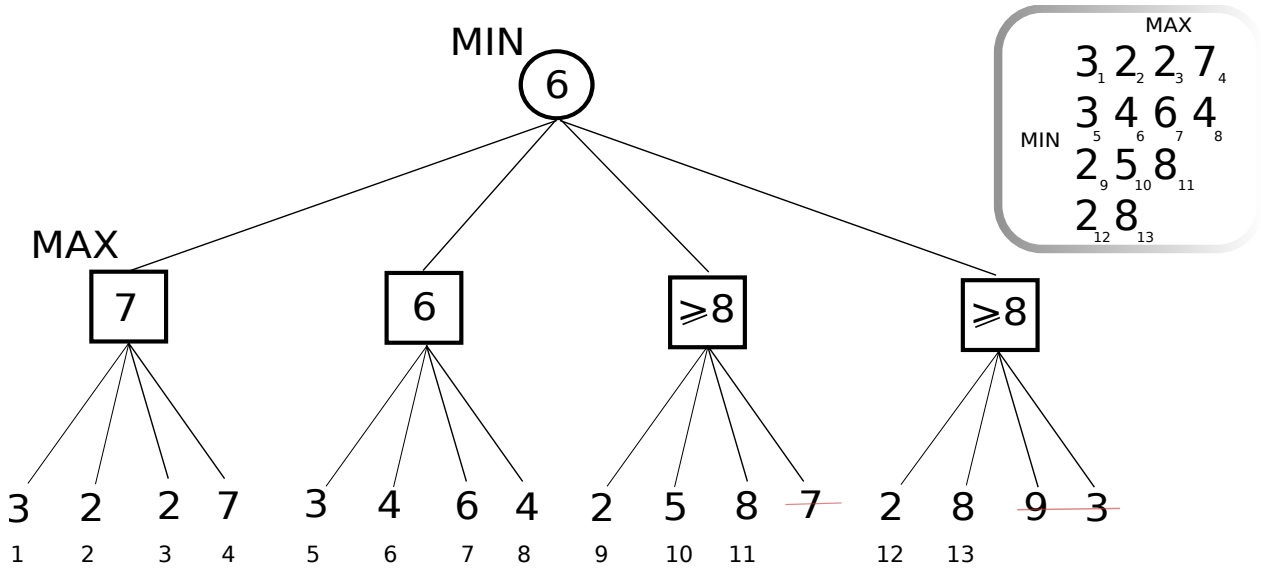


Figure 5.6: Minimax search with alpha pruning. By the time we get to the third MAX node, β is 6; after step 11, we know that the third MAX node's value is at least 8, and therefore the remaining child does not need to be visited.

database with four items. The complete minimax search takes $O(D^2)$ steps.

However, an exhaustive search of the minimax tree is generally not necessary. We can employ known pruning techniques to reduce the number of costly pairwise regret evaluations at the leaves (which require solving linear programs) and potentially achieve linear (rather than quadratic) performance in practice. Figures 5.6, 5.7, and 5.8 describe and illustrate three pruning techniques: alpha pruning, beta pruning, and combined alpha-beta pruning (Pearl, 1984).

Alpha pruning

Let β be the smallest known *upper* bound on the value $v(n^{MIN})$ of the root MIN node; at the outset, it is initialized to $+\infty$. At any time during the search, we know that the final MMR value cannot be greater than β; β can only decrease in value during the search.

For each MAX node n_i^{MAX} (corresponding to row i in the L matrix), let α_i be the largest known value among its children (equivalently, the largest known value in the i^{th} row of L); all αs are initialized to $-\infty$. At any time during the search, we know that the value of MAX node n_i^{MAX} is at least α_i; αs can only increase in value during the search.

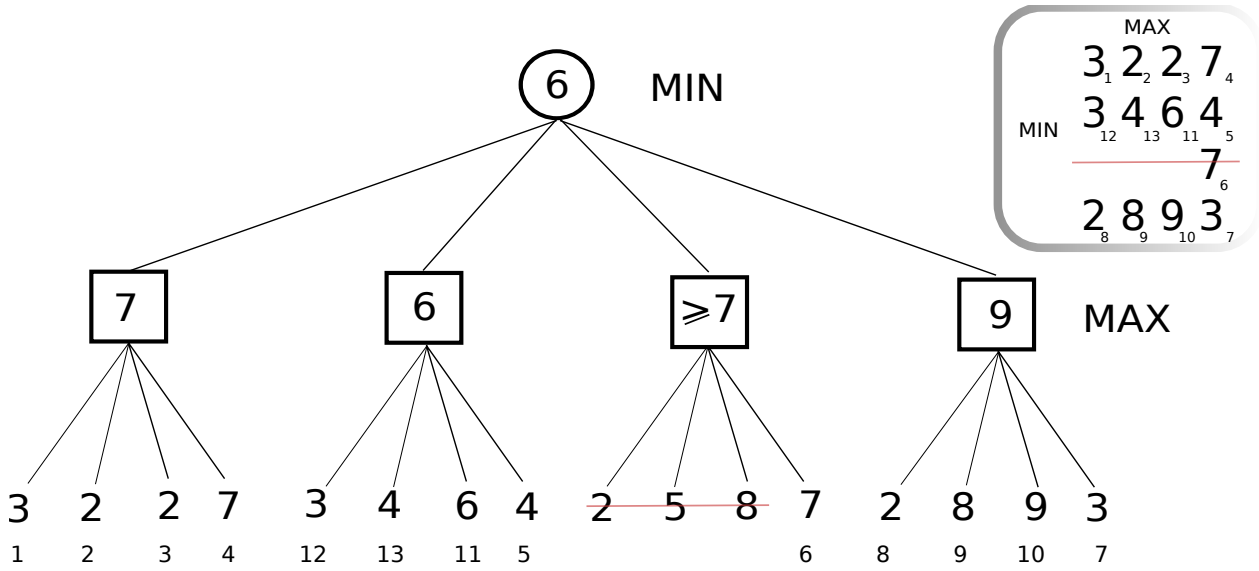


Figure 5.7: Minimax search with beta pruning. By default, the first MIN move is 1, i.e., we explore the first MIN branch/row (steps 1,2,3,4). If MIN chose 1, MAX would counter with move 4, whose value is 7; we therefore set $\beta = 7$ (minimax regret cannot be higher than 7). Now, we search for the next MIN branch to explore by finding the best MIN move to counter the previous MAX move 4 (steps 5,6,7). In the process, we set $\alpha_2 = 4$, $\alpha_3 = 7$, and $\alpha_4 = 3$. The next MIN move is therefore 4, to which MAX would reply with move 3, whose value is 9 (steps 8,9,10). Once again, we search for the best MIN move against the last MAX move 3. Because we already explored MIN moves (rows) 1 and 4, and move 3 can be beta-pruned because $\alpha_3 = 7$ is the same as $\beta = 7$, the last remaining branch to explore is 2 (steps 11,12,13). Its max value is 6, and therefore the minimax value is also 6.

Figure 5.6 shows an example of alpha pruning, during which some MAX node children do not need to be explored. Assume a depth-first traversal of the tree; if, while searching the children of a MAX node n_i^{MAX}, α_i becomes greater than (or equal to) β, we do not need to consider the remaining children of the MAX node. This pruning occurs because, by definition, $\alpha_i \leq v(n_i^{MAX})$, $\beta \geq v(n^{MIN})$; if $\alpha_i \geq \beta$, then $v(n_i^{MAX}) \geq \beta \geq v(n^{MIN})$, and therefore the MAX node n_i^{MAX} will not be chosen by the MIN player. By excluding some terminal nodes from the search, pruning can significantly reduce the MMR computation time, because terminal node evaluations require performing costly LP optimizations.

Beta pruning

Beta pruning affects the children (branches) of the root MIN node. Instead of exploring the MIN node's branches in some predefined order (like depth-first traversal), we pick the next

branch (row) to explore based on some heuristic. In particular, we experiment with what is known in the AI literature as the "killer heuristic" (Akl and Newborn, 1977; Pearl, 1984): choose the branch that minimizes the value with respect to the current MAX choice (which was the best MAX choice with respect to the previous MIN branch). The underlying intuition for the effectiveness of this strategy relies on the hypothesis that in many real-life scenarios, a particular MAX choice (column) will be good against many MIN choices (as we will show below, such heuristic does not work well for randomly generated MMR game instances).

Let j^* be the current MAX "killer" move for MAX node n_i^{MAX} (i.e., with respect to the MIN choice i):

$$j^* = \underset{j \in children(n_i^{MAX})}{\operatorname{argmax}} v(n_{ij}^T).$$

Then, the next branch i' to explore is the one that corresponds to the best MIN choice if MAX chooses j^*:

$$i' = \underset{i \in children(n^{MIN})}{\operatorname{argmin}} v(n_{ij^*}^T).$$

Beta pruning occurs if, when going through all MIN branches to find the next one to explore, some MAX node's α value is greater than β. In that case, the corresponding MIN branch (row) can safely be removed from further consideration. Figure 5.7 shows an example of beta pruning, and describes the steps in more detail.

Alpha-beta pruning

Alpha-beta pruning is like beta pruning (rows are chosen according to a certain heuristic), but the MAX node children are also alpha-pruned. There's a tradeoff between beta and alpha-beta pruning: with no alpha pruning, we get better MAX choices for the heuristic (since we always traverse all children of a MAX node); on the other hand, for each MAX node, alpha-beta pruning explores fewer children, but provides a possibly inferior choice for the MIN node choice heuristic. Experimental results in the next section show very little distinction between beta and alpha-beta pruning. Figure 5.8 shows an example of alpha-beta pruning, and describes the steps in more detail.

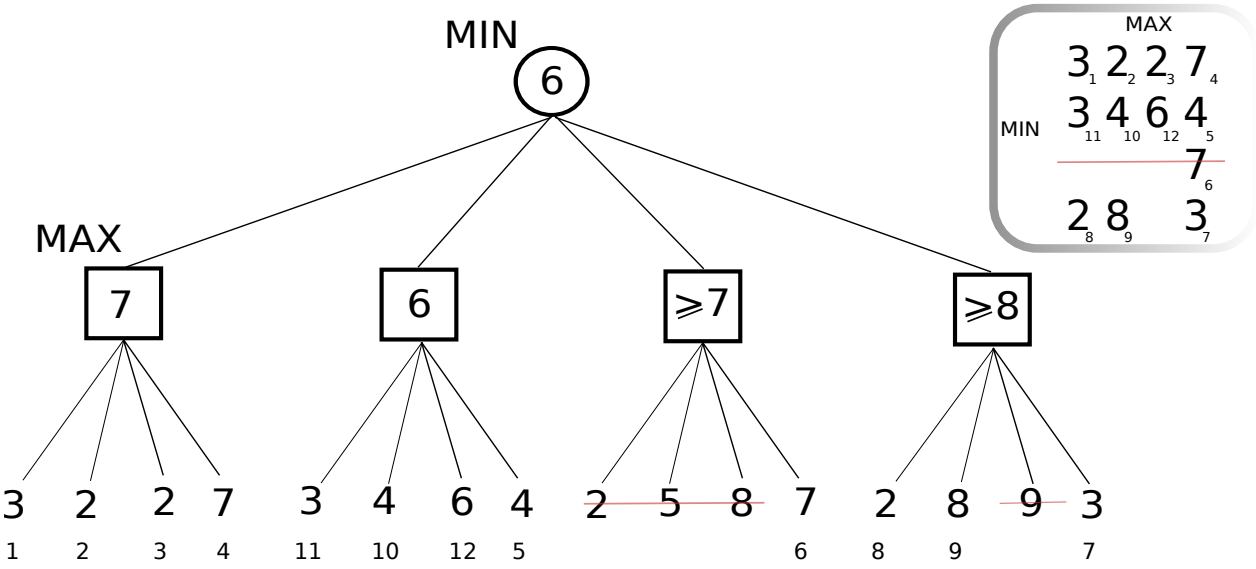
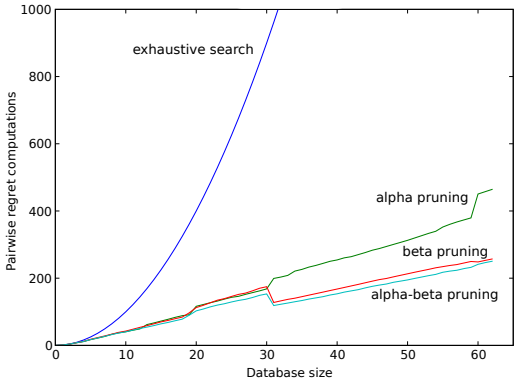
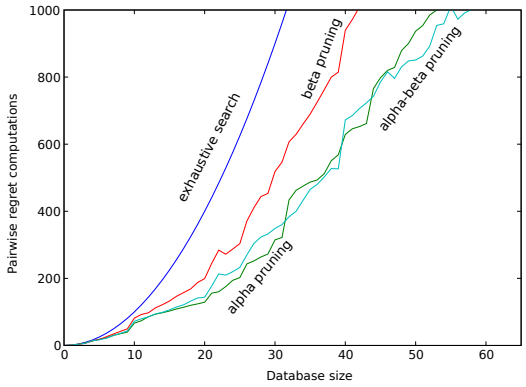


Figure 5.8: Minimax search with alpha-beta pruning. The search steps are very similar to the beta pruning case above. By default, the first MIN move is 1, i.e., we explore the first MIN branch/row (steps 1,2,3,4). If MIN chose 1, MAX would counter with move 4, whose value 7; we set $\beta = 7$ (minimax regret cannot be higher than 7). Now, we search for the next MIN branch to explore by finding the best MIN move to counter the previous MAX move 4 (steps 5,6,7). In the process, we set $\alpha_2 = 4$, $\alpha_3 = 7$, and $\alpha_4 = 3$. The next MIN move is therefore 4, to which MAX would reply with move 2, whose value is 8 (steps 8,9). This differs from the beta pruning case, because MAX now also performs alpha pruning and stops exploring as soon as it finds a leaf node whose value is equal to or larger than β (in this case its value was 8 in step 9). Once again, we search for the best MIN move against the last MAX move 2. Because we already explored MIN moves (rows) 1 and 4, and move 3 can be beta-pruned because $\alpha_3 = 7$ is the same as $\beta = 7$, the last remaining branch to explore is 2 (steps 10,11,12). Its max value is 6, and therefore the minimax value is also 6.



(a) Apartment database



(b) Random matrix. The second-worst curve is beta pruning. Alpha and alpha-beta pruning exhibit almost the same behavior.

Figure 5.9: Minimax search performance on an actual database of 62 apartments (a), and a database with randomly generated pairwise regret values (b). The curves show how the number of pairwise regret evaluations required (each of which is a costly LP optimization) varies with the database size.

The pseudo-code algorithm for the alpha-beta pruning for MMR computation in database problems is provided in the Appendix D.

5.2.4.3 Pruning performance

Figure 5.9(a) shows the impact of pruning during the minimax search of an actual database of 62 apartments described with 8 attributes, and on a database of the same-size with randomly generated pairwise regret values (Figure 5.9(b)). The x axis is the size of the database, ranging from 1 to 62; the y axis plots the number of pairwise-regret evaluations required (each of which is a costly LP optimization) during the minimax search. For each size of the database, the number of search steps is averaged over all possible MIN starting choices (since initial MIN choice is arbitrary, and can heavily influence the search).

The running time of exhaustive search grows quadratically with the database size. However, as shown in Figure 5.9(a), pruning helps a lot, resulting in effectively linear performance on the real database. Beta and alpha-beta pruning work best, verifying the intuition that in structured

Input: Set of feasible outcomes $\mathbf{X}_F$, set of queries Q, set of query costs $\{c_q | q \in Q\}$, sets of responses $\{A_q | q \in Q\}$, sets of prior constraints $\mathcal{G}$ and $\mathcal{U}$ (that specify the initial space of feasible utilities $\mathbf{U}$), query scoring function $S(q|\, \mathbf{U}, \mathbf{X}_F)$, termination criteria T
Output: Recommended outcome $\mathbf{x}^*$ and its max regret $MR(\mathbf{x}^*, \mathbf{U})$

$\mathbf{x}^* \leftarrow \text{argmin}_{\mathbf{x} \in \mathbf{X}_F} MR(\mathbf{x}, \mathbf{U})$
while *termination criteria T not met* **do**
  $q^* = \text{argmax}_{q \in Q} S(q|\, \mathbf{U}, \mathbf{X}_F) - c_q$
  pose query q^* to the user
  receive response a_q
  update the constraints $\mathcal{U}$ (and, therefore, $\mathbf{U}$) based on a_q
  $\mathbf{x}^* \leftarrow \text{argmin}_{\mathbf{x} \in \mathbf{X}_F} MR(\mathbf{x}, \mathbf{U})$
end

Figure 5.10: A generic procedure that supports minimax regret-driven preference elicitation through a sequence of interactions (queries and responses) between the decision support system and the user (decision maker).

problems, there are a few MAX choices that are very good for multiple MIN choices.

On a random problem, pruning is not effective and the running time growth remains quadratic. Also, as expected, beta pruning performs worse than alpha pruning.

5.3 Elicitation

The minimax regret criterion can be used both for making robust decisions under strict uncertainty and for driving the elicitation process. In contrast to the Bayesian elicitation setting, the quality (difference from optimal) of a minimax regret optimal decision can be bounded; these bounds can be tightened with further elicitation effort. At every step of an elicitation process, the system rates available queries according to a scoring function that approximates each query's potential to reduce the minimax regret level. A user's response imposes additional constraints on the feasible utility space $\mathbf{U}$, which results in a new decision situation with a new level of minimax regret (the level of regret cannot increase with more information). The process continues until the minimax regret reaches some acceptable level, elicitation costs (e.g., the number of queries the user has to answer) become too high, or some other condition from

the set T of termination criteria is met.

Figure 5.10 shows the generic minimax-regret driven elicitation framework assumed in this chapter. In this framework, the decision support system interacts with the user by posing queries from the set of available queries Q, and, for each query $q \in Q$, receive user responses (answers) from the set A_q. Each response a_q provides additional information about the feasible utility region $\mathbf{U}$ and leads to an updated feasible utility region $\mathbf{U}^a$. Since responses impose constraints on utility space $\mathbf{U}$, they cannot increase the size of the feasible utility space:

$$\mathbf{U}^a \subseteq \mathbf{U} \text{ for any response } a \in \cup_q A_q. \tag{5.18}$$

This leads to the following observation:

Observation 5.1 *The minimax regret level cannot increase with additional information about user utilities:*

$$MMR(\mathbf{U}^a) \leq MMR(\mathbf{U}) \textit{ for any response } a. \tag{5.19}$$

Proof Because of Eq. 5.18, *for any pair of outcomes* $\mathbf{x}$ *and* $\mathbf{y}$, the pairwise regret under $\mathbf{U}^a$ is the same or lower than under $\mathbf{U}$, which leads to the inequality $MMR(\mathbf{U}^a) \leq MMR(\mathbf{U})$:

$$\begin{aligned}
\mathbf{U}^a \subseteq \mathbf{U} & \iff \\
\max_{u \in \mathbf{U}^a} R_u(\mathbf{x}, \mathbf{y}) \leq \max_{u \in \mathbf{U}} R_u(\mathbf{x}, \mathbf{y}) \text{ for all } \mathbf{x}, \mathbf{y} \in \mathbf{X} & \iff \\
R(\mathbf{x}, \mathbf{y}, \mathbf{U}^a) \leq R(\mathbf{x}, \mathbf{y}, \mathbf{U}) \text{ for all } \mathbf{x}, \mathbf{y} \in \mathbf{X} & \iff \\
\max_{\mathbf{y}} R(\mathbf{x}, \mathbf{y}, \mathbf{U}^a) \leq \max_{\mathbf{y}} R(\mathbf{x}, \mathbf{y}, \mathbf{U}) \text{ for all } \mathbf{x} \in \mathbf{X} & \iff \\
MR(\mathbf{x}, \mathbf{U}^a) \leq MR(\mathbf{x}, \mathbf{U}) \text{ for all } \mathbf{x} \in \mathbf{X} & \iff \\
\min_{\mathbf{x}} MR(\mathbf{x}, \mathbf{U}^a) \leq \min_{\mathbf{x}} MR(\mathbf{x}, \mathbf{U}) & \iff \\
MMR(\mathbf{U}^a) \leq MMR(\mathbf{U}). \ \square &
\end{aligned} \tag{5.20}$$

Response model

Even though the minimax regret decision criterion does not take into account any probabilistic knowledge about the user utilities, such knowledge is potentially very valuable for driving the

elicitation process. A good elicitation policy would ask queries that are effective in reducing minimax regret levels. For each query, we need to consider all possible situations resulting from every possible response. If we do not know how likely each response is, the value of a query can be determined by non-probabilistic response aggregation operators, such as min or max. However, judging the value of a query by assuming the best or worst response often leads to unsatisfactory policies that are either too optimistic or too cautious, and are liable to stall without reducing the minimax regret. An alternative solution is to assume a probabilistic response model, similar to the Bayesian case discussed in the previous chapter. The response model $\Pr(a_q|q,u)$ specifies the probability of receiving the response a_q to the query q under utility function u. If the users always answer accurately, the response model $\Pr(a_q|q,u)$ is a deterministic mapping from a query and utility function to a response a_q. We assume such noiseless responses in our minimax regret based models and practical implementations. If π is the probability distribution over the feasible utility space $\mathbf{U}$, the likelihood of receiving a particular response depends only on π:

$$Pr(a_q|q,\pi) = \mathbb{E}_{u\sim\pi}[(\Pr(a_q|q,u)].$$

In our elicitation strategies, for computational reasons and due to the absence of better prior distributions over utilities, we assume a uniform distribution over feasible utilities.

5.3.1 Myopically optimal strategy (MY)

With sufficient computational resources, one could employ a *myopically optimal* strategy (denoted MY, as in (Boutilier et al., 2006)) which computes exact MMR levels for each response to every query, and chooses the query with the lowest expected post-response MMR level. We can define a myopic score of a query q as the expected reduction in minimax regret:

$$S_{MY}(q) = MMR(\mathbf{U}) - \mathbb{E}_a[MMR(\mathbf{U}^a)], \tag{5.21}$$

where $\mathbf{U}^a$ is the set of feasible utilities resulting from response a to the query q, and expectation is taken with respect to the probabilities of all responses to q. The MY strategy recommends

the query with the highest MY score:

$$q^* = \arg\max\ S_{MY}(q),$$

where q ranges over all possible queries.

The MY strategy is suboptimal, because it greedily selects queries that provide best *immediate* reductions in regret, thus likely missing *sequences* of queries that lead to even better results. By performing a k-step lookahead, this strategy can approximate the sequentially optimal strategy as k increases. However, in practice, even a one-step exact lookahead is prohibitively expensive for problems where real-time interactive elicitation is required.

The MY strategy serves as a good baseline when comparing other strategies, since most other strategies discussed below are approximations of the myopically optimal strategy. Viappiani and Boutilier (2009) explore the application of the MY strategy to *choice queries*, which present the user with a set of options to choose from.[1]

5.3.2 Current solution strategy (CS)

Since computing minimax regret for every response to each possible query is likely infeasible, practical querying strategies rely on heuristic criteria for evaluating potential impact of queries. One effective approach is to concentrate only on utility parameters that are directly involved in the *current solution* of the minimax regret optimization (Boutilier et al., 2004c, 2006; Braziunas and Boutilier, 2007). The current solution triple $\langle \mathbf{x}^*, \mathbf{y}^w, u^w \rangle$, consisting of the regret-minimizing outcome $\mathbf{x}^*$, the adversary's witness $\mathbf{y}^w$, and the utility function u^w chosen by the adversary, defines the minimax regret level $MMR(\mathbf{U}) = u^w(\mathbf{y}^w) - u^w(\mathbf{x}^*)$. Additional constraints on the utility parameters directly involved in determining $u^w(\mathbf{y}^w)$ and $u^w(\mathbf{x}^*)$ are likely to change the MMR level, either by increasing $u^w(\mathbf{x}^*)$ or reducing $u^w(\mathbf{y}^w)$. Focusing on relevant parameters results in elicitation that is directed at reducing minimax re-

[1] Viappiani and Boutilier (2010) extend the myopically optimal choice query optimization to the Bayesian setting.

gret, rather than just overall uncertainty over feasible utilities. This is the main intuition behind current solution based strategies.

The CS score of a query q indirectly measures the query's potential to reduce minimax regret by considering its impact only on the current solution. We define $S_{CS}(q)$ to be the expected change in the *pairwise* regret of the current solution due to a response to the query q:

$$S_{CS}(q) = MMR(\mathbf{U}) - \mathbb{E}_a[R(\mathbf{x}^*, \mathbf{y}^w, \mathbf{U}^a)] = MMR(\mathbf{U}) - \mathbb{E}_a[R_{u^a}(\mathbf{x}^*, \mathbf{y}^w)], \quad (5.22)$$

where

$$u^a = \max_{u \in \mathbf{U}^a} R_u(\mathbf{y}^w, \mathbf{x}^*).$$

That is, the CS score computes the expected reduction in MMR regret *under the constraint that the current solution outcomes* $\mathbf{x}^*$ *and* $\mathbf{y}^w$ *do not change*.

For GAI utilities, every response a adds an additional linear constraint on GAI utility parameters to the constraint set $\mathcal{U}$, resulting in a new constraint set $\mathcal{U}^a$. The pairwise regret $R(\mathbf{x}^*, \mathbf{y}^w, \mathbf{U}^a)$ under $\mathbf{U}^a$ can be found by solving a linear program (Eq. 5.6):

$$R(\mathbf{x}^*, \mathbf{y}^w, \mathbf{U}^a) = \max_{\boldsymbol{\theta}} \sum_{j=1}^{M} \sum_{k=1}^{N_j} (C^k_{\mathbf{y}^w_j} - C^k_{\mathbf{x}^*_j})\, \theta^k_j, \text{ subject to } \mathcal{G} \text{ and } \mathcal{U}^a. \quad (5.23)$$

The CS score $S_{CS}(q)$ for each query is obtained by solving a linear program above for each response a, and then averaging response values weighted by response likelihoods.

While certainly faster than doing a complete one-step lookahead (as in the MY strategy), the CS strategy is still expensive if the space of possible queries and their responses is large, as it requires solving a linear program for every response to every query. In the following section, we present a heuristic scoring function that incorporates further approximations designed to speed up the query selection computation.

5.3.3 UTPREF strategy (UT)

The UTPREF (UT) strategy is an approximation of the CS strategy that is both effective in reducing MMR level and fast to compute. It is a heuristic strategy employed by the UTPREF

recommendation system (described in Chapter 6) that is designed specifically for GAI utilities and limited to the query types listed in Section 3.3. The UT scoring function is much faster to evaluate as it avoids having to solve a set of linear programs for each query.

Consider the minimax regret $MMR(\mathbf{U})$ in a GAI model, defined by the solution triple $\langle \mathbf{x}^*, \mathbf{y}^w, u^w \rangle$ (Eq. 5.6):

$$\begin{aligned} MMR(\mathbf{U}) &= u^w(\mathbf{x}^w) - u^w(\mathbf{x}^*) \\ &= \sum_{j=1}^{M} \sum_{k=1}^{N_j} (C^k_{\mathbf{y}^w_j} - C^k_{\mathbf{x}^*_j})\ \theta^k_j \\ &= \sum_{j=1}^{M} \sum_{k=1}^{N_j} D^k_j \theta^k_j, \end{aligned} \tag{5.24}$$

where $D^k_j = C^k_{\mathbf{y}^w_j} - C^k_{\mathbf{x}^*_j}$, and θ^k_j are GAI parameters that define the witness utility function u^w:

$$u^w(\mathbf{x}) = \sum_{j=1}^{M} \sum_{k=1}^{N_j} C^k_{\mathbf{x}_j}\, \theta^k_j.$$

In the equation $MMR(\mathbf{U}) = \sum_{j=1}^{M} \sum_{k=1}^{N_j} D^k_j \theta^k_j$, each GAI parameter θ^k_j contributes $D^k_j \theta^k_j$ to the MMR level. A response to a query can result in an additional constraint on the parameter θ^k_j, changing the value of the product $D^k_j\ \theta^k_j$, and, potentially, the MMR level as well.

Upper and lower parameter bounds At any point during an elicitation process, the possible values of utility parameters are specified by the set of linear constraints $\mathcal{U}$. We denote the lower and upper bounds for a utility parameter θ^k_j as $\lfloor \theta^k_j \rfloor$ and $\lceil \theta^k_j \rceil$:

$$\begin{aligned} \lfloor \theta^k_j \rfloor &= \min_{\boldsymbol{\theta}} \theta^k_j, \text{ subject to } \mathcal{G} \text{ and } \mathcal{U}, \\ \lceil \theta^k_j \rceil &= \max_{\boldsymbol{\theta}} \theta^k_j, \text{ subject to } \mathcal{G} \text{ and } \mathcal{U}. \end{aligned} \tag{5.25}$$

Such bounds can be computed by solving two simple linear programs *for each parameter*.

If such computations are too expensive, lower and upper bounds for all parameters can be

approximated by solving two linear programs for *all parameters* at once:

$$\min_{\boldsymbol{\theta}} \sum_j \sum_k \theta_j^k, \text{ subject to } \mathcal{G} \text{ and } \mathcal{U},$$
$$\max_{\boldsymbol{\theta}} \sum_j \sum_k \theta_j^k, \text{ subject to } \mathcal{G} \text{ and } \mathcal{U}. \tag{5.26}$$

The solution to the first program approximates lower bounds for all parameters at once; similarly, the solution to the second program approximates the upper bounds. Since approximate bounds are guaranteed to lie within the real parameters bounds $[\lfloor\theta_j^k\rfloor, \lceil\theta_j^k\rceil]$, the approximate bounds are "safe" to use in bound queries. As long as bound queries ask about the point within approximate (or real) bounds, the user cannot provide an inconsistent response.

Halve-the-largest-gap heuristic We refer to the difference between the upper and lower bound of some utility parameter as its *gap*:

$$gap(\theta_j^k) = \lceil\theta_j^k\rceil - \lfloor\theta_j^k\rfloor. \tag{5.27}$$

If we could pose bound queries to the user about any GAI utility parameter, a promising approach would be to query about the mid-point of the gap of a parameter for which the quantity $|D_j^k| gap(\theta_j^k)$ is the largest. Let $\lfloor\theta_j^k\rfloor \leq b \leq \lceil\theta_j^k\rceil$ be the bound for the query "Is $\theta_j^k \geq b$?", denoted as $q_B[\theta_j^k, b]$. Without probabilistic prior information about user utilities, we make a simplifying assumption that the probability of a positive response is $Pr(yes|q_B[\theta_j^k, b]) = \frac{\lceil\theta_j^k\rceil - b}{gap(\theta_j^k)}$, and the probability of a negative response is $Pr(no|q_B[\theta_j^k, b]) = \frac{b - \lfloor\theta_j^k\rfloor}{gap(\theta_j^k)}$. As we noted before, in the equation $MMR(\mathbf{U}) = \sum_{j=1}^{M} \sum_{k=1}^{N_j} D_j^k \theta_j^k$, each GAI parameter θ_j^k contributes $D_j^k \theta_j^k$ to the MMR level. If D_j^k is positive, then we make a simplifying assumption that $\theta_j^k = \lceil\theta_j^k\rceil$, since otherwise $MMR(\mathcal{U})$ could be increased by increasing the value of θ_j^k. Because of constraints $\mathcal{U}$ tying together many different parameters, such assumption is not always correct; however, it is practical enough for the heuristic UTPREF query strategy. Similarly, if D_j^k is negative, then we assume that $\theta_j^k = \lfloor\theta_j^k\rfloor$. If D_j^k is positive, a negative response to the query $q_B[\theta_j^k, b]$ would decrease the contribution of the parameter θ_j^k by $D_j^k\,(\lceil\theta_j^k\rceil - b)$, since now b is the upper bound

on θ_j^k. A positive response would not change anything, since the current value of θ_j^k, chosen by the adversary, is already set to $\lceil\theta_j^k\rceil$. As with MY and CS, the expected change in $MMR(U)$ is the score of the bound query $q_B[\theta_j^k, b]$:

$$\begin{aligned} S_{UT}(q_B[\theta_j^k, b]) &= Pr(yes|q_B[\theta_j^k, b])\ 0 + Pr(no|q_B[\theta_j^k, b])\ D_j^k\ (\lceil\theta_j^k\rceil - b) \\ &= (1 - \frac{\lceil\theta_j^k\rceil - b}{gap(\theta_j^k)})\ D_j^k\ (\lceil\theta_j^k\rceil - b). \end{aligned}$$

If D_j^k is negative, then the opposite holds true. A negative response would not change anything, since the current value of θ_j^k, chosen by the adversary, is already set to the lowest bound $\lfloor\theta_j^k\rfloor$. A positive response to the query $q_B[\theta_j^k, b]$ would further decrease the contribution of the parameter θ_j^k by $|D_j^k|\,(b - \lfloor\theta_j^k\rfloor)$, by raising the lower bound of θ_j^k to b. The score of the bound query $q_B[\theta_j^k, b]$ when D_j^k is negative is:

$$\begin{aligned} S_{UT}(q_B[\theta_j^k, b]) &= Pr(yes|q_B[\theta_j^k, b])\ |D_j^k|(b - \lfloor\theta_j^k\rfloor) + Pr(no|q_B[\theta_j^k, b])\ 0 \\ &= \left(1 - \frac{b - \lfloor\theta_j^k\rfloor}{gap(\theta_j^k)}\right)\ |D_j^k|\,(b - \lfloor\theta_j^k\rfloor). \end{aligned}$$

Because b is a continuous bound, ranging from $\lfloor\theta_j^k\rfloor$ to $UB\theta_j^k$, there are an infinite number of bound queries. However, under the assumptions stated above, the query score is maximized when b is the middle of the gap, both when D_j^k is positive and when D_j^k is negative.

Observation 5.2 *The bound query score $S_{UT}(q_B[\theta_j^k, b])$ is maximized when b is the middle of the $gap(\theta_j^k)$.*

Proof When D_j^k is positive, $S_{UT}(q_B[\theta_j^k, b]) = (1 - \frac{\lceil\theta_j^k\rceil - b}{gap(\theta_j^k)})\ D_j^k\ (\lceil\theta_j^k\rceil - b)$. Taking the derivative with respect to b, we obtain

$$\frac{dS_{UT}(q_B[\theta_j^k, b])}{db} = -D_j^k(1 - \frac{2}{gap(\theta_j^k)}(\lceil\theta_j^k\rceil - b).$$

By setting it to zero, and solving for b, we get $b = \lceil\theta_j^k\rceil - \frac{gap(\theta_j^k)}{2} = 0.5(\lfloor\theta_j^k\rfloor + \lceil\theta_j^k\rceil)$. Similarly, when D_j^k is negative, solving for b also leads to $b = 0.5(\lfloor\theta_j^k\rfloor + \lceil\theta_j^k\rceil)$. Thus, the score of a bound query is maximized when the query point b^* is the middle of the gap. The score

itself is $S_{UT}(q_B[\theta_j^k, b^*]) = 0.5\ |D_j^k|\ \frac{\lfloor\theta_j^k\rfloor+\lceil\theta_j^k\rceil}{2}$, since with probability 0.5, the contribution of θ_j^k decreases by half of the gap times $|D_j^k|$. □

The bound queries that we just discussed involve arbitrary GAI parameters $\theta_j^k = u(\mathbf{b}^{j,k})$ (or, more directly, arbitrary *global* basic outcomes $\mathbf{b}^{j,k}$). We would not normally directly pose such bound to users, because they might be difficult to answer. One exception is anchor bound queries (ABQs) that are a subset of general bound queries; here, the basic outcomes are special in a sense that they are the best and the worst outcomes in each factor (i.e., anchors). However, the general bound queries are useful for analysis and intuitive understanding of the heuristic UT strategy. Such queries are roughly equivalent to the bound queries described by Boutilier et al. (2006), where they are used in a similar way in an HLG (halve-the-largest-gap) elicitation strategy.

The UT strategy, employed by the UTPREF recommendation system, is based on similar heuristic evaluations of other query types, viz., local bound, anchor bound, local comparison and anchor comparison query types. In all cases, the main intuition is the same as for the bound queries discussed above: consider the GAI parameters that define the current MMR level $MMR(\mathbf{U}) = \sum_{j=1}^{M}\sum_{k=1}^{N_j} D_j^k\theta_j^k$, and choose queries that are best at potentially reducing the quantity $\sum_{j=1}^{M}\sum_{k=1}^{N_j} D_j^k\theta_j^k$, with an expectation that the heuristically evaluated impact of a query on this quantity is positively correlated with its actual impact on reducing the minimax regret. In spirit, the UT strategy is similar to the CS strategy. However, it employs more approximations and heuristics (e.g., ignoring constraints $\mathcal{U}$ and $\mathcal{G}$ and using loose parameter bound approximations) to achieve much faster computational efficiency. Next, we describe the scoring functions for different types of queries.

5.3.3.1 Scoring anchor bound queries (ABQs)

Anchor bound queries (introduced in Section 3.3.2.1) involve global factor *anchors* $\mathbf{b}^{j,\top}$ and $\mathbf{b}^{j,\bot}$, i.e., the best and worst outcomes with all attributes outside the factor F_j fixed at their

reference levels. A query $q_{AB}[\theta_j^\top, b]$ asks the user to specify whether the anchor utility $\theta_j^\top = u(\mathbf{b}^{j,\top})$ is greater than the specified bound b (similarly for $\mathbf{b}^{j,\perp}$). By limiting the bound queries to anchor outcomes we can arguably make such queries easier to answer, since most attributes (i.e., attributes outside the given factor) are fixed at reference levels and the attributes inside the factor are set to either best or worst levels.

Since ABQs are a special case of general bound queries considered above, we could simply score them using the "largest gap" heuristic described above:

$$S_{UT}(q_B[\theta_j^{\top/\perp}, b^*]) = 0.5\ |D_j^{\top/\perp}|\ \frac{\lfloor \theta_j^{\top/\perp} \rfloor + \lceil \theta_j^{\top/\perp} \rceil}{2},\ \text{for all } j = 1..M.$$

However, by its nature, the scoring function above dismisses all parameters whose coefficients D_j^k are zero in the equation $MMR(\mathbf{U}) = \sum_{j=1}^{M} \sum_{k=1}^{N_j} D_j^k \theta_j^k$. Since the number of parameters with zero coefficients (including anchor parameters) is large, for many anchors their bound query score will often be zero. Treating anchor outcomes like any regular basic outcome undervalues their importance in defining (and potentially reducing) the MMR level. Anchor outcomes are special for two reasons:

a) all factor parameter values are bounded by the top and bottom anchor values, and therefore an additional constraint on an anchor parameter is likely to impact many other parameters:

$$\theta_j^\perp \leq \theta_j^k \leq \theta_j^\top \text{ for all } k = 1..N_j, \text{for all } j = 1..M;$$

b) and, local bound queries impose constraints of the form $\theta_j^i - b\theta_j^\top - (1-b)\theta_j^\perp \geq 0$, which further tie anchor parameters to other factor parameters.

Thus, anchor parameters are quite influential in the constraint set $\mathcal{U}$. A better way of scoring ABQs uses the local value parameterization of GAI utilities.

A local value function $v_j(\mathbf{x}_j^k)$ is related to the GAI parameters θ_j^k through the following transformation (Section 3.2.3):

$$v_j^k = v_j(\mathbf{x}_j^k) = \frac{\theta_j^k - \theta_j^\perp}{\theta_j^\top - \theta_j^\perp},$$

where an LVF parameter v_j^k denotes the value $v_j(\mathbf{x}_j^k)$. The MMR equation can be rewritten as follows:

$$\begin{aligned}
MMR(\mathbf{U}) &= \sum_{j=1}^{M}\sum_{k=1}^{N_j}(C^k_{\mathbf{y}_j^w} - C^k_{\mathbf{x}_j^*})\ \theta_j^k \\
&= \sum_{j=1}^{M}(\theta_j^\top - \theta_j^\perp)\sum_{k=1}^{N_j}(C^k_{\mathbf{y}_j^w} - C^k_{\mathbf{x}_j^*})\ \frac{\theta_j^k}{\theta_j^\top - \theta_j^\perp} \\
&= \sum_{j=1}^{M}(\theta_j^\top - \theta_j^\perp)\sum_{k=1}^{N_j}(C^k_{\mathbf{y}_j^w} - C^k_{\mathbf{x}_j^*})\ (\frac{\theta_j^k}{\theta_j^\top - \theta_j^\perp} - \frac{\theta_j^\perp}{\theta_j^\top - \theta_j^\perp}) \\
&= \sum_{j=1}^{M}(\theta_j^\top - \theta_j^\perp)\sum_{k=1}^{N_j}(C^k_{\mathbf{y}_j^w} - C^k_{\mathbf{x}_j^*})\ \frac{\theta_j^k - \theta_j^\perp}{\theta_j^\top - \theta_j^\perp} \\
&= \sum_{j=1}^{M}(\theta_j^\top - \theta_j^\perp)\sum_{k=1}^{N_j}(C^k_{\mathbf{y}_j^w} - C^k_{\mathbf{x}_j^*})\ v_j^k \\
&= \sum_{j=1}^{M}(\theta_j^\top - \theta_j^\perp)\ r_j(\mathbf{x}_j^*, \mathbf{y}_j^w), \qquad (5.28)
\end{aligned}$$

where $r_j(\mathbf{x}_j, \mathbf{y}_j)$ can be thought of as unscaled local factor regret of choosing $\mathbf{y}_j$ rather than $\mathbf{x}_j$:

$$r_j(\mathbf{x}_j, \mathbf{y}_j) = \sum_{k=1}^{N_j}(C^k_{\mathbf{y}_j} - C^k_{\mathbf{x}_j})\ v_j^k. \qquad (5.29)$$

The equality between the second and third line in Eq. 5.28 holds because, for any $\mathbf{x} \in \mathbf{X}$, $\sum_{k=1}^{N_j} C^k_{\mathbf{x}_j} = 0$ if $j \neq 1$, and $\sum_{k=1}^{N_1} C^k_{\mathbf{x}_1} = \theta_1^\perp$ for $j = 1$ (this was proved in Observation 3.3 before). Therefore, for any $j \in 1..M$,

$$\begin{aligned}
\sum_{k=1}^{N_j}(C^k_{\mathbf{y}_j^w} - C^k_{\mathbf{x}_j^*})\frac{\theta_j^\perp}{\theta_j^\top - \theta_j^\perp} &= \frac{\theta_j^\perp}{\theta_j^\top - \theta_j^\perp}\ \left(\sum_{k=1}^{N_j} C^k_{\mathbf{y}_j^w} - \sum_{k=1}^{N_j} C^k_{\mathbf{x}_j^*})\right) \\
&= \frac{\theta_j^\perp}{\theta_j^\top - \theta_j^\perp}\ 0 \\
&= 0.
\end{aligned}$$

By rewriting Eq. 5.28 as

$$MMR(\mathbf{U}) = \sum_{j=1}^{M} r_j(\mathbf{x}_j^*, \mathbf{y}_j^w)\,\theta_j^\top + \sum_{j=1}^{M} -r_j(\mathbf{x}_j^*, \mathbf{y}_j^w)\,\theta_j^\perp, \qquad (5.30)$$

and comparing to

$$MMR(\mathbf{U}) = \sum_{j=1}^{M} \sum_{k=1}^{N_j} D_j^k \theta_j^k,$$

we can see that the local regrets $r_j(\mathbf{x}_j^*, \mathbf{y}_j^w)$ play the same role for anchor parameters as difference coefficients D_j^k for generic GAI parameters. Furthermore, in Eq. 5.30, the anchor coefficients $r_j(\mathbf{x}_j^*, \mathbf{y}_j^w)$ are better suited to reflect the importance of anchor outcomes, and are less likely to be zero (the local regret $r_j(\mathbf{x}_j^*, \mathbf{y}_j^w)$ is zero only if $\mathbf{x}_j^* = \mathbf{y}_j^w$; in such a case, it does make intuitive sense not to query about the anchor bounds for the factor F_j). The local regrets $r_j(\mathbf{x}_j^*, \mathbf{y}_j^w)$ can be easily computed given the parameters θ_k^j for the adversary's utility function choice u^w. By analogy to the general bound queries discussed above (i.e., substituting $r_j(\mathbf{x}_j^*, \mathbf{y}_j^w)$ for D_j^k) we can see that the best query bound b^* is always the midpoint of the gap, and the best anchor to query is the one with the largest quantity $|r_j(\mathbf{x}_j^*, \mathbf{y}_j^w)| gap(\theta_j^{\top,\perp})$. The UT scoring function for ABQs is:

$$S_{UT}(q_{AB}[\theta_j^{\top/\perp}, b^*]) = 0.5 \; |r_j(\mathbf{x}_j^*, \mathbf{y}_j^w)| \; \frac{\lfloor \theta_j^{\top/\perp} \rfloor + \lceil \theta_j^{\top/\perp} \rceil}{2}, \text{ for all } j = 1..M. \tag{5.31}$$

Due to the nature of ABQ queries, even if reducing uncertainty about anchor parameters does not lead to an immediate reduction in the MMR level, it is still likely that constraining influential anchor parameters will provide benefits a few steps later in the elicitation process.

5.3.3.2 Scoring local bound queries (LBQs)

A *bound query* asks the user to consider a single outcome, and decide whether its value is greater or less than some specified bound b. In a *local bound query* (LBQ, introduced in Section 3.3.1.3) $q_{LB}[v_j^k, b]$, the outcome is a local factor outcome $\mathbf{x}_j^k$ whose LVF $v_j(\mathbf{x}_j^k)$ is between 0 (if $\mathbf{x}_j^k$ is the worst factor outcome $\mathbf{x}_j^\perp$) and 1 (if $\mathbf{x}_j^k = \mathbf{x}_j^\top$).

The UT score of an LBQ is based on the same principles as the score for general bound and anchor bound queries. To see the how much a given local outcome contributes to the MMR

level, we rewrite Eq. 5.28 as follows:

$$\begin{aligned} MMR(\mathbf{U}) &= \sum_{j=1}^{M}\sum_{k=1}^{N_j}(C^k_{\mathbf{y}^w_j} - C^k_{\mathbf{x}^*_j})\ \theta^k_j \\ &= \sum_{j=1}^{M}(\theta^\top_j - \theta^\perp_j)\sum_{k=1}^{N_j} D^k_j\ \frac{\theta^k_j - \theta^\perp_j}{\theta^\top_j - \theta^\perp_j} \\ &= \sum_{j=1}^{M}(\theta^\top_j - \theta^\perp_j)\sum_{k=1}^{N_j} D^k_j\ v^k_j \\ &= \sum_{k=1}^{N_j}(\theta^\top_j - \theta^\perp_j)D^k_j\ v^k_j + \sum_{j'\neq j}\sum_{k=1}^{N_{j'}} D^k_{j'}\ \theta^k_{j'}. \end{aligned} \tag{5.32}$$

In this equation, each local value parameter for factor F_j contributes $(\theta^\top_j - \theta^\perp_j)D^k_j\, v^k_j$ to the overall MMR level. Thus, among all local bound queries for factor F_j, the UT strategy picks the one associated with the local configuration $\mathbf{x}^k_j$ with the largest quantity $(\theta^\top_j - \theta^\perp_j)|D^k_j|\, gap(v^k_j)$. Just as for general bound and anchor bound queries, the optimal bound to ask about is the middle of the gap (Observation 5.2). This leads to the following scoring function for LBQ queries:

$$S_{UT}(q_{LB}[v^k_j, b^*]) = 0.5\ (\theta^\top_j - \theta^\perp_j)|D^k_j|\ \frac{\lfloor v^k_j \rfloor + \lceil v^k_j \rceil}{2},\ \text{for all } j = 1..M. \tag{5.33}$$

Computing local bounds

One complicating difference between local and global bound queries is the difficulty of determining the lower and upper bounds $\lfloor v^k_j \rfloor$ and $\lceil v^k_j \rceil$ for each LVF parameter v^k_j. For global parameters θ^k_j, their bounds can be found by solving simple linear programs, with constraints $\mathcal{U}$ and $\mathcal{G}$ (see Section 5.3.3). For LVF parameters, computing the bounds is more problematic. An LVF parameter v^k_j is related to GAI parameters as follows:

$$v^k_j = \frac{\theta^k_j - \theta^\perp_j}{\theta^\top_j - \theta^k_j}.$$

Therefore, solving for $\max v^k_j$ (or $\min v^k_j$), subject to linear constraints $\mathcal{U}$ on global parameters $\{\theta^k_j\}$, is a non-linear optimization problem. In practice, we use loose local bounds obtained

from global parameter bounds and bounds computed from constraints imposed directly on local value parameters by previous local bound and comparison queries. From the equation above, since we know the bounds on $\theta_j^k, \theta_j^\top, \theta_j^\perp$, we obtain:

$$\lfloor v_j^k \rfloor \geq \frac{\lfloor \theta_j^k \rfloor - \lceil \theta_j^\perp \rceil}{\lceil \theta_j^\top \rceil - \lceil \theta_j^\perp \rceil},$$
$$\lceil v_j^k \rceil \leq \frac{\lceil \theta_j^k \rceil - \lfloor \theta_j^\perp \rfloor}{\lfloor \theta_j^\top \rfloor - \lfloor \theta_j^\perp \rfloor}.$$

In addition to the constraint set $\mathcal{U}$ of constraints on global parameters θ_j^k, we maintain a separate constraint set $\mathcal{U}_v$ of linear constraints on LVF parameters v_j^k obtained from local bound and local comparison queries. With $\mathcal{U}_v$, we can compute loose bounds on v_j^k by solving linear programs $\min v_j^k$ and $\max v_j^k$, subject to $\mathcal{U}_v$. Since $\mathcal{U}_v$ only includes information from responses to local queries, the bounds obtained are not exact. By combining the two types of bounds, however, we get a reasonable approximation of the true local bounds, which is good enough in practice for the query strategy. We should note that with more complex constraints on utility parameters from different types of queries, it is possible that the query bound (i.e., the middle of the estimated gap) might fall outside the true gap, resulting in a response that is inconsistent with the previous preference information summarized by the constraint set $\mathcal{U}$. In practice, once a query is selected, we test for inconsistency of the updated constraint $\mathcal{U}$ for each possible binary response, and reject the query if it leads to inconsistent responses; by updating the local value constraint set $\mathcal{U}_v$, we also ensure that the same query does not get selected in the future.

5.3.3.3 Scoring comparison queries (ACQs and LCQs)

A basic outcome comparison query $q_C[\theta_{j_1}^{k_1}, \theta_{j_2}^{k_2}]$ asks the user to compare two basic outcomes $\mathbf{b}^{j_1,k_1}$ and $\mathbf{b}^{j_2,k_2}$. The response imposes an inequality constraint between the two corresponding GAI parameters $\theta_{j_1}^{k_1}$ and $\theta_{j_2}^{k_2}$. As described in Section 3.3, both local comparison queries (LCQs) and anchor comparison queries (ACQs) are basic outcome queries.

In an LCQ, both outcomes are local outcomes that belong to the same factor; in addition, the user is asked to assume that the attributes in the factor's conditioning set are fixed at reference levels. A sample query would be: "Assume that the attributes in K_j are fixed at reference levels. Would you prefer the local outcome $\mathbf{x}_j^{k_1}$ to the local outcome $\mathbf{x}_j^{k_2}$, ceteris paribus?" If the answer is "yes", then $v_j(\mathbf{x}_j^{k_1}) \geq v_j(\mathbf{x}_j^{k_2})$, and, therefore, $\theta_j^{k_1} \geq \theta_j^{k_2}$; if the answer is "no", then $\theta_j^{k_1} \leq \theta_j^{k_2}$.

In an ACQ, both outcomes to be compared are either top or bottom anchors for some (different) factors. Just like LCQs, ACQs are a subset of basic outcome comparison queries, which result in inequality constraints on two GAI parameters.

$$\mathbf{b}^{j_1,\top} \succeq \mathbf{b}^{j_2,\top} \iff \theta_{j_1}^{\top} \geq \theta_{j_2}^{\top}. \tag{5.34}$$

To simplify the notation and unify the presentation for both LCQs and ACQs, let's assume that every local configuration in every factor can be identified a single global index. That is, we assume a one-to-one mapping between any pair $\langle j, k \rangle$ (where j is the factor index, and k is the local configuration index) and a global index p.[1] In this way, each GAI utility parameter can be addressed by a single index:

$$\theta_j^k = \theta_p.$$

A GAI utility function can then be written as

$$u(\mathbf{x}) = \sum_p C(\mathbf{x})_p \, \theta_p,$$

where $C(\mathbf{x})_p = C_{\mathbf{x}_j}^k$ is the structure coefficient for the parameter θ_p. The MMR equation

[1] For example, we could set $p = \sum_{j'=1}^{j-1} N_{j'} + k$ to compute the global index of the parameter θ_j^k (N_j is the number of local configurations in factor F_j).

becomes:

$$\begin{aligned} MMR(\mathbf{U}) &= \sum_{j=1}^{M} \sum_{k=1}^{N_j} (C^k_{\mathbf{y}^w_j} - C^k_{\mathbf{x}^*_j})\ \theta^k_j \\ &= \sum_{p} (C(\mathbf{x}^*)_p - C(\mathbf{y}^w)_p)\ \theta_p \\ &= \sum_{p} D_p\ \theta_p, \end{aligned}$$

where $D_p = C(\mathbf{x}^*)_p - C(\mathbf{y}^w)_p$.

Let P be the set of indices of parameters that are being considered for a comparison query. For LCQs, P indexes the set of all basic outcomes for a particular factor F_j. For ACQs, P specifies the set of all anchor parameters. The final list of comparison queries to be scored is obtained by considering all pairs of parameters $\{(\theta_{p^1}, \theta_{p^2}) | p^1, p^2 \in P\}$ involved in the current solution such that:

(a) $D_{p^1} \neq 0$ and $D_{p^2} \neq 0$ for all $p^1, p^2 \in P$;

(b) $\lceil \theta_{p^1} \rceil \geq \lfloor \theta_{p^2} \rfloor$ and $\lceil \theta_{p^2} \rceil \geq \lfloor \theta_{p^1} \rfloor$; and,

(c) the relationship between θ_{p^1} and θ_{p^2} is not known due to earlier queries.

The first condition limits the pairs of parameters to be considered to those that actually contribute to the current MMR level due to non-zero coefficients D_p. The second condition checks the bounds for implied relationships, because if one parameter's lower bound is greater than the other's upper bound, a comparison query would be meaningless. The third (which subsumes the second) makes sure that the relationship between the two parameters is not already known beforehand due to transitive closure of previous comparison constraints (or, more generally, due to prior constraints $\mathcal{U}$ or $\mathcal{G}$).

For that purpose, we maintain a *preorder* data structure that keeps track of order relationships among all utility parameters, marking a preference relationship between each pair of parameters as $\succeq$, $\preceq$, or ? (unknown). If a new relationship is added, the data structure triggers an iteration through all related parameter pairs, updating unknown relationships according to

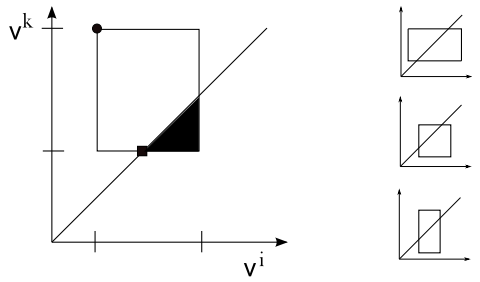


Figure 5.11: Four different ways to bisect a bounding rectangle. The shaded area approximates the feasible parameter space after a response to a comparison query. In all cases, if the response to a comparison query eliminates the part of the rectangle which contained the current solution point $(\theta_{p^1}, \theta_{p^2})$ (marked with a circle), the new solution point (marked with a square) is one of the two intersections of the diagonal and the bounding rectangle. v^i corresponds to θ_{p^1}, and v^k corresponds to θ_{p^2}.

transitive preference rules. The preorder data structure provides an efficient way to filter out the pairs of parameters with known preference relationships. However, it is possible that, due to complex linear constraints in $\mathcal{U}$ resulting from bound and general comparison queries, the preorder data structure deems a preference relationship between a pair of parameters as unknown, when in fact, due to linear constraints $\mathcal{U}$, the relationship is fixed. For computational efficiency, we use the preorder data structure to score different pairs of parameters, but, once the optimal pair selected for the query, we verify that the query would not result in an inconsistent response by solving two linear programs, subject to constraints $\mathcal{U}$ and $\mathcal{G}$. For the two parameters θ_{p^1} and θ_{p^2}, if $\max \theta_{p^1} - \theta_{p^2} \leq 0$, then $\theta_{p^2} \succeq \theta_{p^1}$ under all feasible utilities in $\mathbf{U}$, so the pair of parameters should not be queried (but their relationship should be added to the preorder data structure). Similarly, if $\max \theta_{p^2} - \theta_{p^1} \leq 0$, then $\theta_{p^1} \succeq \theta_{p^2}$. In all other cases, the preference relationship between θ_{p^1} and θ_{p^2} is unknown (given $\mathbf{U}$), so the comparison query can be asked.

To score comparison queries, we approximate the feasible utility parameter space $\mathbf{U}$ with a bounding hyperrectangle. For each pair of parameters $(\theta_{p^1}, \theta_{p^2})$, we compute a heuristic score as follows. First, we project the bounding hyperrectangle on the plane of the two parameters; the comparison constraint divides the 2-D projection along the 45-degree line. Fig. 5.11 shows four possible cases and demonstrates that, after a response to a comparison query, the values of the parameters $(\theta_{p^1}, \theta_{p^2})$ (as well as the current level of regret) either remain the same, or they

are pushed to lie at one of the two intersections of the diagonal with the bounding rectangle. In the latter case, the reduction in regret can be approximated by

$$D_{p^1}\theta_{p^1} + D_{p^2}\theta_{p^2} - \max(D_{p^1}t_1 + D_{p^2}t_1, D_{p^1}t_2 + D_{p^2}t_2), \tag{5.35}$$

where (t_1, t_1) and (t_2, t_2) are the coordinates of the two intersections. The probability of the regret-changing response is estimated by comparing the areas above and below the 45-degree line (see Fig. 5.11).

The heuristic score S_{UT} weighs the approximated reduction in regret (computed by the equation above) by the probability of the regret-changing response.

5.3.3.4 Scoring global comparison queries (GCQs and GCPQs)

Global comparison queries (GCQs) are the most general comparison queries. Rather than limiting the two outcomes in the query to basic outcomes (or a subset of basic outcomes, such as anchor outcomes, or basic outcomes in the same factor), in a GCQ query, the two outcomes to be compared are arbitrary global outcomes. Because there is an exponential number of GCQ queries, and because an arbitrary GCQ query can be difficult to reason about, the only GCQ queries that we consider are current solution comparison queries $q_C[\mathbf{x}^*, \mathbf{y}^w]$, in which the two outcomes are the MMR-optimal choice $\mathbf{x}^*$ and the adversary's choice $\mathbf{y}^w$.

In certain domains, such as renting an apartment or choosing a travel package, outcomes have an associated *price*, which we treat as a special attribute, measured in monetary units. We make the standard assumption of *quasilinear utility* in which, overloading u, the utility $u(\mathbf{x}, p)$ of an outcome $\mathbf{x}$ obtained at price p is $u(\mathbf{x}, p) = \alpha u(\mathbf{x}) - p$. Here $u(\mathbf{x})$ is the price-independent utility of $\mathbf{x}$ and α is a valuation factor that adjusts u for currency (to keep things simple, we assume that $\alpha = 1$). A GCQ that asks a user to compare two current solution outcomes $\mathbf{x}$ and $\mathbf{y}$ while taking into account their price attributes $p_{\mathbf{x}}$ and $p_{\mathbf{y}}$ is called the *global comparison query with price* (GCPQ). If the price attribute is not considered when comparing the two outcomes $\mathbf{x}$ and $\mathbf{y}$, the query is simply a GCQ.

As with all the queries, a response to a GCQ or a GCPQ imposes a linear constraint on GAI parameters, potentially constraining many parameters at once:

$$\begin{aligned}(\mathbf{x}, p_{\mathbf{x}}) \succeq (\mathbf{y}, p_{\mathbf{y}}) &\iff u(\mathbf{x}, p_{\mathbf{x}}) \geq u(\mathbf{y}, p_{\mathbf{y}}) \\ &\iff \sum_{j=1}^{M} \sum_{k=1}^{N_j} C^k_{\mathbf{x}_j}\ \theta^k_j - p_{\mathbf{x}} \geq \sum_{j=1}^{M} \sum_{k=1}^{N_j} C^k_{\mathbf{y}_j}\ \theta^k_j - p_{\mathbf{y}} \\ &\iff \sum_{j=1}^{M} \sum_{k=1}^{N_j} (C^k_{\mathbf{x}_j} - C^k_{\mathbf{y}_j})\ \theta^k_j \geq p_{\mathbf{x}} - p_{\mathbf{y}}.\end{aligned}$$

Since $C^k_{\mathbf{x}_j}$, $C^k_{\mathbf{y}_j}$, $p_{\mathbf{x}}$ and $p_{\mathbf{y}}$ are known, the resulting inequality is a linear constraint on GAI parameters θ^k_j. For GCQ queries, where the price attribute is not considered, we can assume that $p_{\mathbf{x}} = p_{\mathbf{y}}$.

With price, the current solution outcomes are $(\mathbf{x}^*, p_{\mathbf{x}^*})$ and $(\mathbf{y}^w, p_{\mathbf{y}^w})$. If the user prefers $(\mathbf{x}^*, p_{\mathbf{x}^*})$, then the adversary's choice has to change, since $u(\mathbf{x}^*, p_{\mathbf{x}^*}) \geq u(\mathbf{y}^w, p_{\mathbf{y}^w})$ for all $u \in \mathbf{U}^a$, making the pairwise regret $R((\mathbf{x}^*, p_{\mathbf{x}^*}), (\mathbf{y}^w, p_{\mathbf{y}^w}), \mathbf{U}^a) = R(\mathbf{x}^*, \mathbf{y}^w, \mathbf{U}^a) + (p_{\mathbf{x}^*} - p_{\mathbf{y}^w})$ for the current solution zero. If the user actually prefers the adversary's choice $(\mathbf{y}^w, p_{\mathbf{y}^w})$, then the current solution choices $(\mathbf{x}^*, p_{\mathbf{x}^*})$ and $(\mathbf{y}^w, p_{\mathbf{y}^w})$ are not guaranteed to change; however, in practice, they often do, because of an additional constraint on $\mathcal{U}$.

For GCPQs, where the current solution outcomes include price, there is no obvious way to compute heuristic scores that would be well-calibrated against scores for other query types (since GCPQs are very "powerful" queries, almost guaranteed to change the current solution). Therefore, we usually employ GCPQs separately, without combining them with other types of queries. GCQs, however, can be incorporated in our standard scoring-based elicitation framework. The user is asked to compare the two current solution outcomes *while ignoring the special price attribute*. For a heuristic score S_{UT} for a GCQ $q_C[\mathbf{x}^*, \mathbf{y}^w]$, we use the S_{CS} score,

which can be computed by solving two linear programs (Eq. 5.22):

$$
\begin{aligned}
&S_{UT}(q_C[\mathbf{x}^*, \mathbf{y}^w]) = S_{CS}(q_C[\mathbf{x}^*, \mathbf{y}^w]) \\
&= MMR(\mathbf{U}) - \mathbb{E}_a[R(\mathbf{x}^*, \mathbf{y}^w, \mathbf{U}^a)] \\
&= MMR(\mathbf{U}) - \left(0.5 \max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}_{\mathbf{U}^{yes}}} \sum_{j=1}^{M} \sum_{k=1}^{N_j} (C^k_{\mathbf{y}^w_j} - C^k_{\mathbf{x}^*_j})\, \theta^k_j + 0.5 \max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}_{\mathbf{U}^{no}}} \sum_{j=1}^{M} \sum_{k=1}^{N_j} (C^k_{\mathbf{y}^w_j} - C^k_{\mathbf{x}^*_j})\, \theta^k_j \right).
\end{aligned}
\tag{5.36}
$$

Here $\boldsymbol{\Theta}_{\mathbf{U}^{yes}}$ and $\boldsymbol{\Theta}_{\mathbf{U}^{no}}$ are GAI parameter spaces after "yes" and "no" responses, and we assume that both responses are equally likely (since it is computationally hard to compute exact likelihoods due to the complex shape of the polytope $\boldsymbol{\Theta}_{\mathbf{U}}$).

5.4 Experimental results

In this section, we compare the performance of various elicitation strategies on the apartment rental database problem (Appendix C.2) and the car rental configuration problem (Appendix C.1). The apartment rental problem comprises a database of 200 apartments, described by ten attributes (including price), each having between two and four domain values. The GAI model has eight factors. The car rental problem is modeled with 26 attributes that specify various attributes of a car relevant to typical rental decisions. The domain sizes of the attributes range from two to nine values, resulting in 6.1×10^{10} possible configurations. The GAI model consists of 13 local factors, each defined on at most five variables; the model has 378 utility parameters. There are ten hard constraints defining feasible configurations. Appendix C contains full descriptions of the domains.

The algorithms were implemented in Python; ILOG CPLEX 9.1 was to used to solve LP and MIP optimizations. For both problems, computing the regret-minimizing solution, which has to updated after each query, takes less than one second; determining the next query for any elicitation strategy also takes less than one second. Thus our approach admits real-time interaction.

Combining different queries

We use all six types of basic queries in our experiments: LB (local bound), LC (local comparison), AB (anchor bound), AC (anchor comparison), GC (current solution global comparison), and GCP (GC with price). During elicitation, each query has a score S_{UT} computed using techniques described in Section 5.3.3. If all types of queries are available, we can simply choose the next query to ask based on the heuristic score S_{UT}. However, in general we want to consider not only the impact of a query in reducing regret, but also its cost to a user. Global queries might be harder to answer than local queries; similarly, most users will find comparing two outcomes easier than dealing with bound queries. Therefore, we investigate the performance of several "qualitative" strategies that combine different query types without explicitly differentiating for cost.

- **LC** *LC strategy* uses only local comparison queries. Elicitation terminates when no local comparison queries can be used to further reduce minimax regret.
- **LC(LB)** If instead of terminating when no LCQ has a positive score, we select the best local bound query, we get the *LC(LB) strategy*.
- **LB** *LB strategy* uses only local bound queries.
- **GCP** *GCP strategy* employs only current solution comparison queries (which include the price attribute). Guided by the MMR optimal solution, this strategy uses pairwise comparisons of outcomes to find the best one. GCP strategy is only applicable in the apartment rental domain (since configurations in the car rental domain do not include price). It is the only strategy that does not use the heuristic scoring function S_{UT}.

The remaining strategies do not favor any query type, but simply select a query with the highest score from the set of allowed query types:

- **LB+LC** combines local comparison and bound queries,

	LB	LC	AB	AC	GC	GCP
LC	0	100	0	0	0	0
LC(LB)	71	29	0	0	0	0
LB	100	0	0	0	0	0
LB+LC	73	27	0	0	0	0
AB+AC	0	0	86	14	0	0
LB+AB	49	0	51	0	0	0
LC+AC	0	66	0	34	0	0
LB+LC+AB+AC+GC	24	9	27	7	34	0
GCP	0	0	0	0	0	100

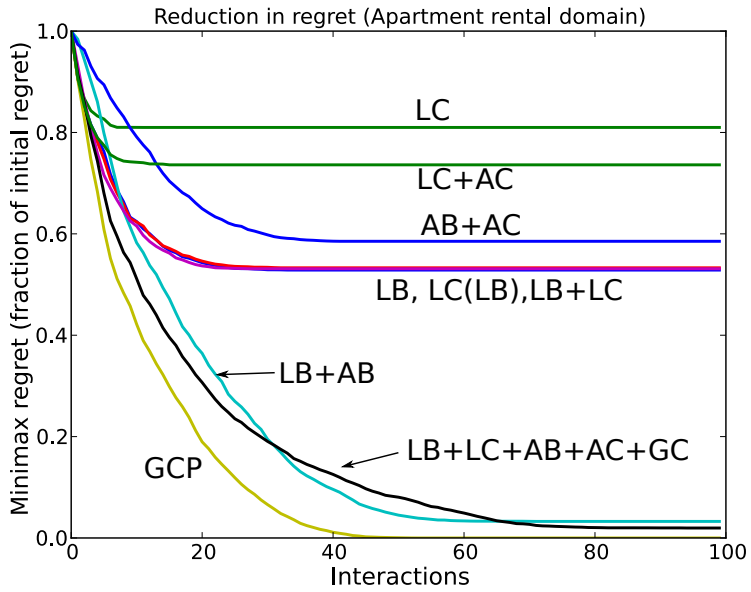
Table 5.3: The table lists empirical query type percentage proportions for different query strategies, averaged over 100 random instantiations of user utilities in the apartment rental domain. Each elicitation session consists of up to 100 interactions.

- **AB+AC** uses only global queries,
- **LB+AB** uses only bound queries,
- **LC+AC** uses only comparison queries, and
- **LB+LC+AB+AC+GC** or **all query** strategy uses all available queries (except GCP).

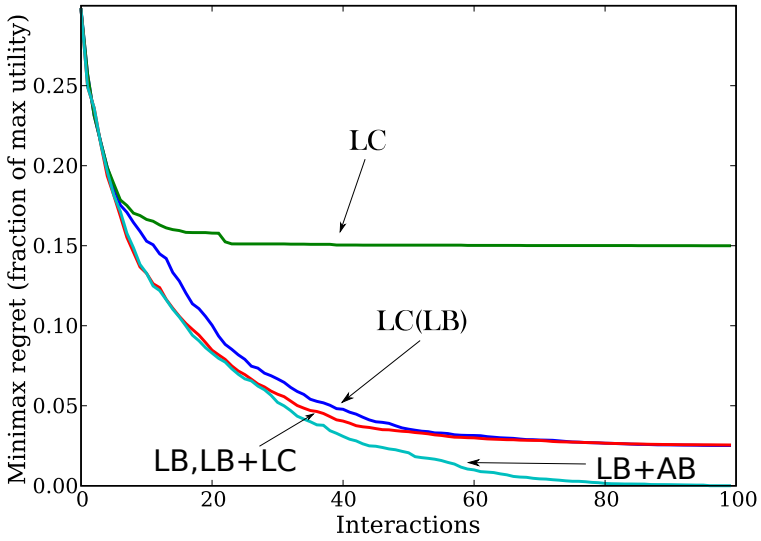
To get a sense of how frequently different queries are asked using these different strategies, Table 5.3 lists empirical query type proportions for different query strategies, averaged over 100 random instantiations of user utilities in the apartment-rental domain.

Results

Fig. 5.12(a) shows the performance of the nine query strategies described above on the apartment rental database problem; Fig. 5.12(b) shows the performance of five strategies on the car rental configuration problem. Elicitation sessions terminate after 100 queries. The results are averaged over 100 random instantiations of user utilities for the apartment rental problem, and

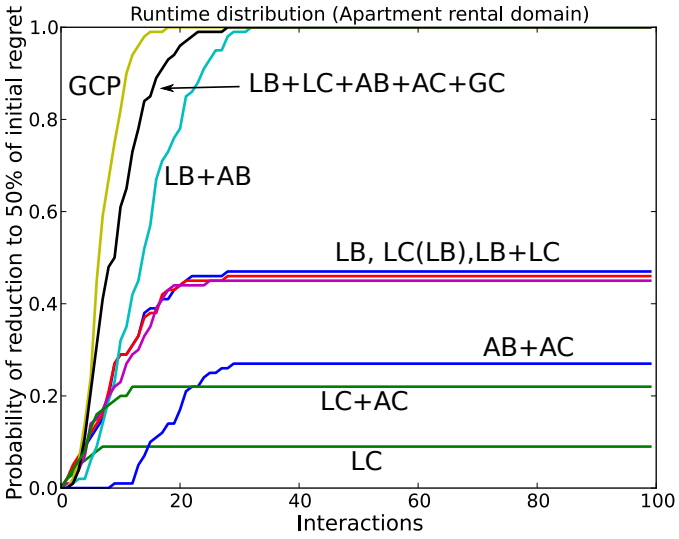


(a) Apartment rental problem

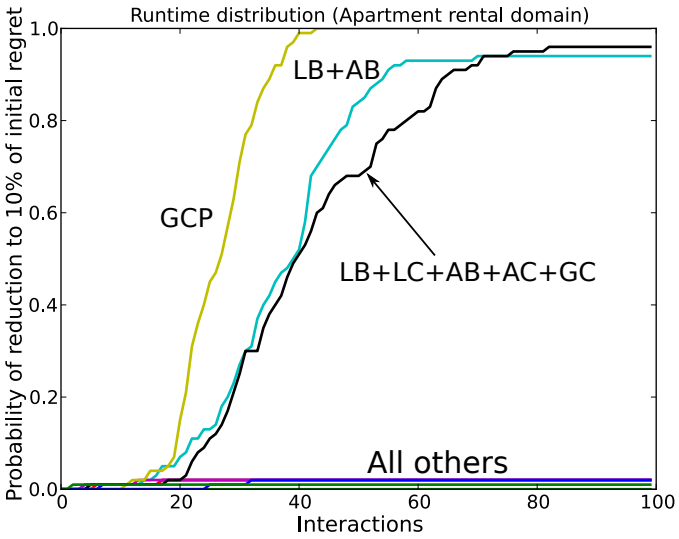


(b) Car rental problem

Figure 5.12: The performance of different query strategies on the (a) apartment rental database problem; (b) the car rental configuration problem. Each curve shows how the minimax regret is reduced as the number of query-response interactions increases. The figure (b) for the car rental configuration domain is taken from (Braziunas and Boutilier, 2007); the experiments for the configuration domains did not include GCP and "all queries" strategies (although, from (a), we can expect that "all queries" strategy performs very similar to the LB+AB strategy).



(a) Runtime distribution - 50% of initial regret



(b) Runtime distribution - 10% of initial regret

Figure 5.13: Runtime distributions of different query strategies in the apartment rental domain. Each curve represents the probability that the strategy achieves the reduction to (a) 50% of initial regret, and (b) 10% of initial regret, for a given number of elicitation queries (interactions).

20 instantiations for the car rental problem. In the following discussion, we concentrate on the apartment rental domain (Fig. 5.12(a)), but strategy performance follows similar trends in the car rental domain as well.

Fig. 5.12(a) shows how the average minimax regret is reduced (starting at 100% of initial

regret) with additional utility information obtained via preference queries. The nine strategies can be roughly partitioned into two groups based on their performance. The first group contains strategies that are not powerful enough to achieve significant reduction in minimax regret after the first 20 or 30 steps. It includes LC, LC+AC, AB+AC, LB, LC(LB), and LB+LC strategies; the performance of LB, LC(LB), and LB+LC strategies is virtually identical. While natural and easy to answer, comparison (both local and global) queries do not provide enough quantitative information on their own to effectively reduce uncertainty over user utilities. Local bound queries are more effective, but are of limited impact if used without the cross-factor calibration provided by global queries.

The second group contains three best-performing strategies: LB+AB, LB+LC+AB+AC+GC, and GCP. All the strategies in this group reduce the minimax regret to close-to-zero after approximately 50 queries. The competitive performance of the LB+AB strategy demonstrates the necessity of both LB and AB queries in elicitation strategies. The LB+LC+AB+AC+GC (also known as the *all query*) strategy successfully combines five different query types based on the S_{UT} scoring function without sacrificing performance. On average, it uses 24% LBQs, 9% LCQs, 27% ABQs, 7% ACQs, and 34% GCQs (see Table 5.3 for query proportions for other query strategies). Finally, the GCP strategy that uses only pairwise comparison queries involving the two current solution outcomes (with price) turns out to be both the most effective and the simplest of all.

Figure 5.13 provides a different evaluation of query strategy performance by showing the runtime distributions of different query strategies in the apartment rental domain. Each curve represents the probability that the strategy achieves the reduction to (a) 50% of initial regret, and (b) 10% of initial regret, for a given number of elicitation queries. Figure 5.13(b) clearly shows the separation of the nine query strategies into two groups defined above, since the strategies in the first, less effective, group do not have any chance of reducing the minimax regret to 10% of its initial value.

With the exception of the AB+AC strategy, all strategies (including those that use only local

queries) exhibit a sharp initial reduction in minimax regret (by around 20% after the first 10 queries or so). This means that in many cases we can initially use local comparison queries (which are generally less costly in terms of user effort, time and accuracy than bound queries), and then switch to using bound queries for further reduction of regret. In domains with a relatively small number of attributes (where the user can comfortably compare two arbitrary outcomes) GCPQs perform very well. With more attributes, global comparison queries become costlier, making local queries (especially LBQs) more attractive.

In the next chapter, we describe a user study that evaluates the performance of GCP and "all queries" strategy with real users.

5.5 Conclusion

Minimax regret was first recognized by Boutilier et al. (2001); Salo and Hämäläinen (2001) as a robust decision criterion under utility function uncertainty. It is used by Wang and Boutilier (2003) with flat (unstructured) utilities, and by Boutilier et al. (2004c) with additive utilities. Most relevant to the material in this chapter is the work by Boutilier et al. (2001, 2003b, 2005, 2006) on applying minimax regret to decision making with and elicitation of GAI utility models. In this chapter, we extend this work by addressing several limitations of the previous approaches. First, we show how to incorporate *semantically sound* local queries into GAI preference elicitation framework, and how to compute minimax regret in configuration problems using the parametric representation of GAI models. Second, we also consider database domains, and apply minimax search pruning techniques to speed-up regret computation in databases domains. For elicitation, we extend the current solution heuristic idea to derive scores for *all* types of queries described in Sec 3.3; specifically, the incorporation of *comparison* queries into the framework is of great practical significance, since such queries are very natural for users to understand and respond to (see Chapter 6).

There are several directions for future work on minimax regret-driven elicitation of factored utility models. Minimax regret is a robust decision criterion, providing decision quality guarantees for any possible realization of user utility function. In some domains, it is desirable to use the minimax regret criterion for final recommendation, even though prior probabilistic information about user utilities is available. We could take advantage of probabilistic information about user utilities in optimizing elicitation strategy, but still use the robust MMR decision criterion for the final decision (Wang and Boutilier, 2003). For example, for bound queries, we could optimize the query bound using the Bayesian value-of-information criterion, rather than simply setting the query bound to the mid-point of the bound interval. A related issue is consideration of optimal (non-myopic) query strategies. As discussed in the previous chapter, computing a sequentially optimal elicitation policy is an intractable problem, equivalent to solving a large continuous-space POMDP. However, it might be possible to devise scoring heuristics that provide a better approximation of a query's sequential (even of limited lookahead horizon) value.

In this chapter, we considered several types of queries that are well grounded in terms of decision-theoretic semantics and simple enough to be used in actual interactive applications with human subjects. However, there is a variety of additional preference query types and modes of interaction that can be effective in eliciting human preferences. Examples include graphical sliders to provide the bounds on utility parameters, or a set of items to choose the preferred item from (Viappiani and Boutilier, 2009, 2010). For each new query type, we have to provide a user interface, incorporate it into our modeling framework, and devise good heuristic scoring functions for use in mixed query strategies.

Successful implementation of decision support systems requires consideration of both descriptive and prescriptive approaches to the preference elicitation problem. In addition to the sound decision-theoretic framework, we also need to address human-centred issues, such as framing and ordering effects, sensitivity analysis and robustness, and the reliability and acceptability of different modes of interaction (Pu et al., 2003). Such empirical validation is only

obtainable with actual user studies, which can make our interaction models more natural and understandable for end users, as well as more effective in providing good recommendations. The next chapter describes a user study that is the first step in this direction.

Chapter 6

UTPREF system and user study

Contents

6.1 Introduction

In this chapter, we describe the first full-fledged user study evaluating the benefits of minimax regret-driven elicitation and recommendation (Braziunas and Boutilier, 2010).

We first introduce the UTPREF Recommendation System, a fully implemented system that helps students navigate and find rental accommodation (or apartments) from a university housing database. UTPREF assumes a multiattribute, *generalized additive utility model* of a student's preferences over housing features, and asks queries of several distinct forms about these preferences. Responses impose constraints on the parameters of the student's utility model, and the system uses the minimax regret decision criterion for several purposes: to recommend a rental unit at any point in the interaction; to assess the quality of the recommended unit, specifically bounding how far it is from the (unknown, user-specific) optimal apartment in the database; and to select queries that have the greatest potential to improve the recommendation. UTPREF exploits minimax regret to "prove" that the optimal apartment has been found with very limited information about the user's underlying utility function.

We then present the results of a study in which 40 participants used the UTPREF system under a variety of conditions to find their most preferred apartment from the database. We assess how far the UTPREF-recommended apartment is from the true optimum (using a method discussed below), measure the effectiveness of regret-based elicitation in finding (near-) optimal products, and evaluate user understanding of the minimax regret criterion and its recommendations. Since UTPREF uses GAI utility models, which are more flexible, but also more complex than commonly used additive models, we are also interested in their value: do they provide for higher quality decisions in practice than an additive approximation. We also assess whether decision-theoretically valid local queries for GAI models allow for more accurate estimation of preferences than simpler queries that ignore the conditioning context (i.e., the assumption that,

in local queries, attributes in the factor conditioning set are to be assumed fixed at reference values). Finally, we provide some assessment of the relative difficulty of specific query types for users.

6.2 UTPREF recommendation system

The UTPREF recommendation system is a software tool that explicitly models user preferences with a GAI utility function, incrementally acquires preference information through a sequence of queries and responses, and recommends a minimax regret-optimal option to the user. It is designed to work with multiattribute database domains, although a suitably modified recommendation system could support configuration problems as well. In particular, in its current implementation, UTPREF provides recommendations for rental accommodation (apartments) in Toronto using a university housing database. Each accomodation unit is described by nine attributes in addition to its price (monthly rent): area, building type, number of rooms, furnished or not, availability of laundry, parking, dishwasher, storage room, and central air conditioning. We make the standard assumption of *quasilinear utility* in which, overloading u, the utility $u(\mathbf{x}, p)$ of an outcome $\mathbf{x}$ obtained at price p is $u(\mathbf{x}, p) = u(\mathbf{x}) - p$; here, $u(\mathbf{x})$ is the price-independent utility of $\mathbf{x}$. Price (rent) is, therefore, a special attribute. In the database of 100 apartments, sampled from the University of Toronto housing database, rents range from \$500 to \$1800; area is divided into four geographic regions; building type (house, apartment, basement) and number of rooms has three values; the remaining attributes are binary. Utility structure (i.e., GAI factors) is configurable; in the user study described in this chapter, we used a common fixed structure with two intersecting factors with two attributes in each: *area, building type* and *building type, number of bedrooms*; the remaining factors had single attributes.

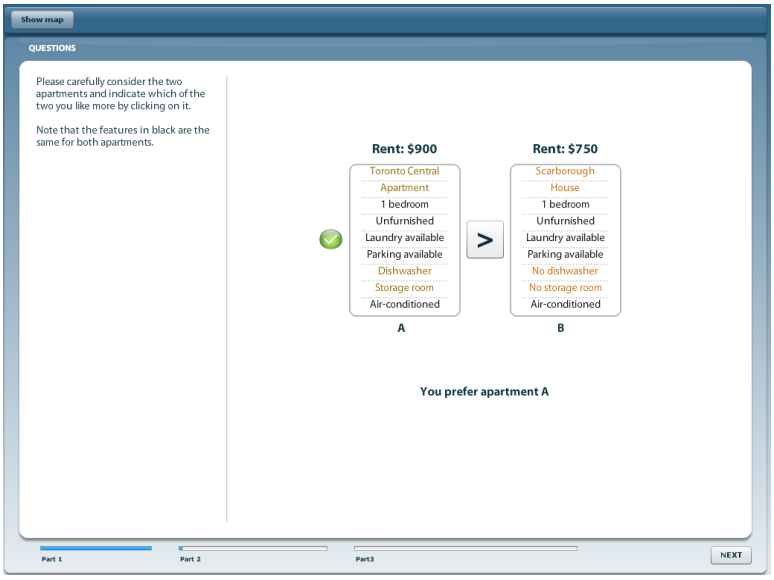


Figure 6.1: Screenshot from a UTPREF session

6.2.1 Implementation

UTPREF consists of a back-end recommendation engine, and a graphical user interface. The recommendation engine is written in Python and uses ILOG CPLEX for the solution of linear programs needed to compute pairwise regret. The graphical user interface is written in Adobe Flex and runs inside the Adobe Flash Player either as a stand-alone cross-platform application or a web browser plugin (see Fig. 6.1 for a sample screenshot of the UTPREF session).

A recommendation session proceeds in two stages. In the preliminary stage A, the system assesses basic user preference information. First, the user is asked to configure a *reference outcome* using drop-down lists of values for each attribute. For example, in the apartment domain, it could be the apartment in which the user currently resides or aspires to reside. In the UTPREF user interface, the exact instruction reads "Please configure a *reference apartment*. This should be an apartment that you are most familiar with. A good choice is an apartment you currently live in, or an apartment you would like to move into." The reference outcome attribute values feature in many types of elicitation queries; such queries are, in fact, tailored to the user based on the selected reference outcome. In local queries, the attributes in the factor conditioning set are fixed to reference values; in global anchor queries, *all* the attribute values

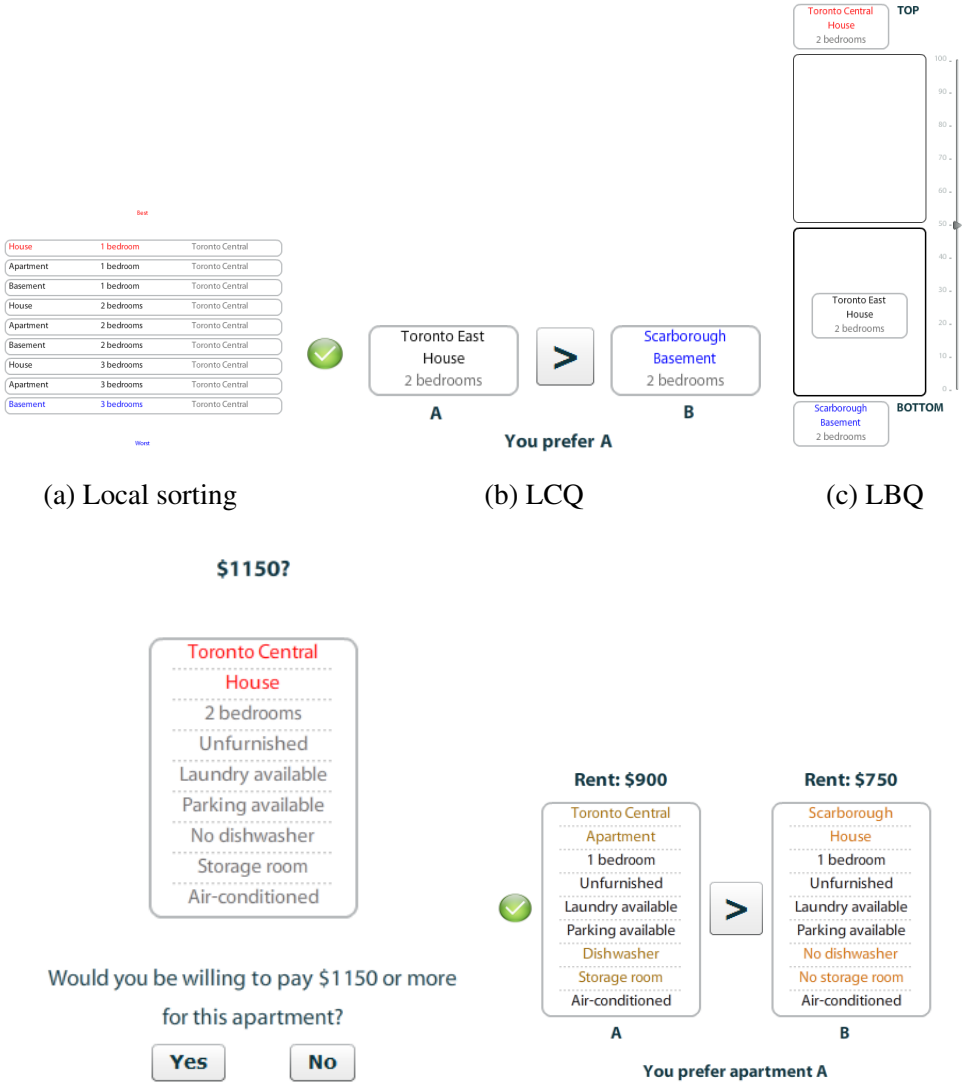
that are outside the query factor are fixed to reference values. Therefore, choosing a familiar reference outcome makes it easier for the user to answer most types of elicitation queries.

After configuration of the reference outcome, for each GAI factor, the user is shown a list of the factor's local outcomes, together with the conditioning set attributes fixed to reference values, and is asked to *sort* that list in terms of decreasing preference via a drag-and-drop interface (see Fig. 6.2(a)). This sorted list provides both the best and the worst factor outcomes (which are used later in local bound queries), as well as a complete ordering of the (local) factor outcomes. Obtaining the full ordering of factor outcomes is easy when the number of factor configurations is relatively small (in the housing domain used in the user study, the largest number of outcomes to sort was nine), since the users are generally familiar with graphical sorting interfaces. The task of sorting N factor outcomes provides $O(N^2)$ implicit pairwise comparisons and therefore saves us from having to ask $O(N^2)$ local comparison queries.

The preliminary preference information acquired in stage A is rarely enough for the user to accept the recommended (MMR-optimal) decision, because its maximum regret is generally still too high. In stage B, the system asks the user a series of queries, until minimax regret drops to an acceptable level. UTPREF employs five different types of queries, described below, and can be configured to implement a variety of elicitation strategies that differ in which query types are used, and what criteria are used to choose a query. At each step, the user can opt to see the MMR-optimal recommendation and its max regret level, and then decide whether to continue with further elicitation, or accept the system's recommendation.

6.2.2 Query types

UTPREF employs several types of GAI queries that can be classified as either *comparison* or *bound* queries; queries are further distinguished by the type of outcomes involved (*local* or *global*) (see Section 3.3). All query types have a precise decision-theoretic semantics dictated by the theory of generalized additive independence, and responses to any query impose linear constraints on the GAI model parameters. An equally important characteristic of these query



(a) Local sorting (b) LCQ (c) LBQ

(d) ABQ (e) GCPQ

Figure 6.2: Five different query types

types is their user-friendliness and usefulness in practical applications, as corroborated by the user study results reported in Section 6.4.

Local queries

Local queries involve only a small subset of all attributes, namely attributes in some GAI factor and the attributes in the factor's conditioning set; the values of remaining attributes do not matter. In UTPREF, we use three types of local queries: comparison, sorting, and bound queries.

A *comparison query* asks the user to compare two outcomes and select the more preferred one. In a *local comparison query* (Fig. 6.2(b)), both outcomes are local outcomes and belong to the same factor. In addition, the user is asked to assume that the attributes in the factor's conditioning set are fixed at reference levels. A response to an LCQ induces in a simple inequality constraint between two GAI utility function parameters.

Sorting a list of factor outcomes determines all pairwise relationships between factor outcomes (i.e., it is similar to asking a large set of pairwise LCQs). Fig. 6.2(a) shows an example of the sorting interface for a factor with `ApartmentType` and `NumberOfBedrooms` attributes; notice that the `Area` attribute, which is the local conditioning set, is fixed at the same reference value for all outcomes (users are alerted to this fact elsewhere on the screen).

A *bound query* asks the user to consider a single outcome, and decide whether its value is greater or less than some specified bound b. In UTPREF, we approximate the probabilistic semantics of LBQs by asking the user to simply consider a local outcome $\mathbf{x}_j$ (in factor F_j) on a scale from 0 (which is the local value of the worst factor outcome $\mathbf{x}_j^{\perp}$) to 100 (which is the local value of the best factor outcome $\mathbf{x}_j^{\top}$), and specify if its value v is greater or less than the bound b. If $v \geq b$, then, according to the LBQ probabilistic semantics, $\mathbf{x}_j \succeq \langle b/100, \mathbf{x}_j^{\top}; 1-b/100, \mathbf{x}_j^{\perp}\rangle$; otherwise, $\mathbf{x}_j \prec \langle b/100, \mathbf{x}_j^{\top}; 1-b/100, \mathbf{x}_j^{\perp}\rangle$. As in other local queries, the attributes in the conditioning set are fixed to reference values. A response to an LBQ imposes linear bound constraints that tie together three different GAI utility parameters.

In the screenshot shown in Fig. 6.2(c), the user has indicated that the value of the *Toronto East, House* outcome is somewhere between 0 and 50 by dragging the outcome in question to the lower bin. The user can also adjust the query "boundary" (in this case 50) by moving the slider to provide tighter or looser constraints if they feel comfortable doing so.[1] Since by adjusting the query boundary the user is more likely to narrow the bounds of the *selected* bin, such user-driven bound adjustment provides stronger restrictions on the feasible utility space. On the other hand, by allowing the user to adjust the bounds, we open up the possibility of inconsistent responses (all other queries are guaranteed not to result in contradictory constraints $\mathcal{U}$ on user utilities). While there are many ways of dealing with inconsistent responses, a straightforward practical approach is to ignore user-adjusted bounds for inconsistent responses.

Global queries

Global queries ask a user to consider preferences over full outcomes. UTPREF uses three types of global queries: anchor bound, anchor comparison and global comparison queries.

Anchor bound queries ABQs involve factor *anchors*, i.e., the best and worst factor outcomes with all attributes outside the factor (not just the conditioning set) fixed at their reference levels. The user only has to specify whether the anchor utility is greater than the specified bound. In UTPREF, we use a monetary scale to calibrate global outcome utilities. In the screenshot in Fig. 6.2(d), the user is simply asked "Would you be willing to pay (at least) $1150 for the specified apartment?"

We also use three variations of *global comparison queries*. GCQs and *GCQs with price* (GCPQs) ask a user to consider two arbitrary global outcomes, ignoring price in the former case, and accounting for price in the latter. An example GCPQ is shown in Fig. 6.2(e). The user is asked to select the better option. A particular form of GCQ is the *anchor comparison query* (ACQ), in which *both* outcomes to be compared are either top or bottom anchors for some

[1] While this boundary adjustment functionality is supported by UTPREF, it was not enabled during the user study (described below).

factor. ACQs are likely easier to understand because, unlike general GCQs, most attributes are fixed at reference levels, which are stable and salient. They also lead to linear constraints that involve only two utility parameters whereas responses to general GCQs or GCPQs tie together multiple utility parameters.

6.2.3 Elicitation strategies

UTPREF system can support a variety of query strategies by restricting the types of queries allowed, and employing different scoring methods. In the user study described below, we test only two strategies due to the relatively small number of participants. The two strategies are both the best-performing on the apartment rental domain with simulated users (see Fig. 5.12(a) and Fig. 5.13 in Section 5.4) and the most diverse in terms of query types and query scoring methods.

The first strategy is the *all query* (also known as LB+LC+AB+AC+GC) strategy, introduced in Section 5.4. It mixes all types of queries except GCPQ (LBQ, LCQ, ABQ, ACQ and GCQ), and at each step selects a query based on the heuristic scoring function S_{UT} described in Section 5.3.3.[1] GCPQs are not included in the *all query* strategy for because they do not have S_{UT} scores, which makes it difficult to meaningfully compare them with other query types in terms of the regret-reduction potential.

On the other end of the spectrum lies the *GCPQ only strategy*. It relies exclusively on GCPQs that ask only about the current solution outcomes (the first one being the minimax optimal outcome, and the second being the adversarial witness; both outcomes include the special price attribute). For this strategy, no query scoring function is needed. If the first choice is preferred, then we can be sure that utility constraints will lead to a different minimax regret solution; otherwise, if the adversary's choice is preferred, additional constraints almost

[1] In the actual user study described in this chapter, information about all local pairwise comparisons was obtained beforehand by asking the users to sort each factor's local outcomes. Therefore, the "all query" strategy did not use local comparison queries. With larger factors, sorting all local outcomes might be too cumbersome, in which case the strategy will potentially employ LCQs as well.

always lead to a different current solution (if the current solution does not change, the strategy defaults to the *all query* strategy; however, we have not encountered this in practice).

6.3 User study design

We conducted a user study with the UTPREF system with three main objectives: 1) to assess overall user comprehension and acceptance of minimax regret-based elicitation; 2) to measure the costs, in terms of time and perceived difficulty, of different query types; and 3) to evaluate the effectiveness of the GAI utility representation as a model of user preferences, investigate the importance of context in local queries in GAI models, and compare different query strategies.

6.3.1 Setup

We recruited 40 participants from the University of Toronto who had either searched for rental housing in Toronto in the previous year, or were considering a new rental in the near future. The primary task of participants involved searching for apartments, using the UTPREF system, from a database of 100 Toronto apartments (sampled from a real Toronto housing database). The number of apartments was chosen to be large enough to justify the use of an intelligent search aid, but small enough for a user to evaluate each option in the second phase of the study, as discussed below. Other related studies have used databases of similar size (Viappiani, Faltings, and Pu, 2006; Portabella Clotet and Rajman, 2006).

Each user session was divided in two parts. In Part 1, users answered preference queries posed by UTPREF which then recommended an apartment based on user responses. In Part 2, users explicitly rated all apartments in the database and ranked the top few options, allowing us to assess the quality of the recommended apartment relative to the user's explicitly stated preferences.

After a brief introduction and a five-minute demonstration, each user session was controlled by the UTPREF system itself, with minimal researcher oversight. Participants were guided by

UTPREF via prompts, task descriptions, and queries. In the first stage of elicitation (Part 1A), participants were asked to configure a reference outcome, and provide an ordering of the local outcomes in each factor using the simple sorting interface (thus also specifying best and worst factor outcomes). In Part 1B, participants responded to a series of local and global queries about their preferences. After ten queries, the system displayed the minimax-optimal apartment and its max regret. The user could then stop the process, or continue answering further queries until minimax regret was reduced to a satisfactory level.

After a short break, participants completed Part 2 of the session, which involved answering a questionnaire about the experience, rating all the apartments in the database, and sorting a small list of apartments in terms of preference. Part 2 was designed to provide an independent assessment of user preferences over the items in the database, in order to evaluate UTPREF 's recommendation quality in Part 1. The most demanding task in Part 2 involved rating all 100 apartments in the database (10 screens with 10 apartments in each) on a 0-2 scale, with score semantics: 0, *I do not like this apartment*; 1, *Not sure*; and 2, *I would rent this apartment*. The rationale behind this task was to quickly identify a set of higly preferred apartments (with the score of 2), without requiring the users to sort the entire list of 100 items. Instead of sorting the entire list, after the rating process, the users were asked to sort (in order of preference) a much smaller set of top-rated apartments, which we call the *final list*. The final list was formed by taking the union of, and randomly shuffling, up to ten highest user-rated apartments from the first task of Part 2, and five apartments with the least max regret as determined in Part 1 (because some apartments were in both sets, the final list could have fewer than 15 elements; for the participants in the UTPREF study, the list sizes ranged from 7 to 12 apartments). The final list always included the recommended outcome, but it was not distinguished in any way.[1] In the last task of Part 2, users specified an approximate *value difference* (in dollars/monthly rent) between the best and worst options in the final list: users were shown their best and worst

[1] Because of the break induced by the questionnaire, participants were less likely to precisely recall the recommended apartment.

options in their sorted list, and asked how much the rent of the best option would have to increase to make the user roughly indifferent between the two options.

6.3.2 Study subgroups

One main feature of UTPREF is the use of more flexible, but more complex, GAI utility functions to represent user preferences. We believe UTPREF is the first preference elicitation system to employ GAI models (rather than additive or unstructured models). To evaluate the benefits and limitations of using this more complex, but semantically sound, model, we tested UTPREF under three main modeling assumptions.

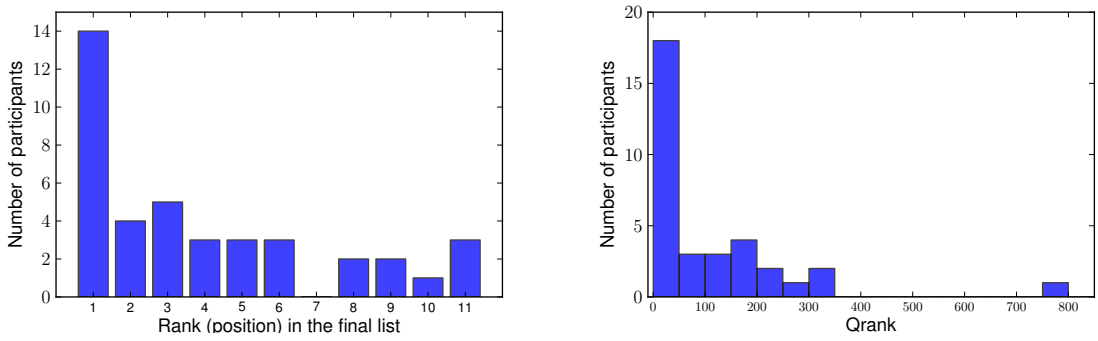
The first condition, *GAI*, uses a full GAI utility model, reflecting preferential dependence among attributes using GAI factors. Elicitation techniques and local queries are those mandated by the theory of GAI modeling (see Chapters 3 and 5). Utility structure (i.e., GAI factors) was fixed for all participants in the study. The utility function had two intersecting factors with two attributes in each: *area, building type* and *building type, number of bedrooms*; the remaining factors had single attributes.[1]

The second condition, *GAI with no local context (GAI-nc)*, was designed to test the sensitivity of GAI elicitation to the use of sound conditioning sets. GAI-nc is identical to GAI, using the same GAI model, except that UTPREF does not display the conditioning context when asking local factor queries (e.g., in the local sorting task or when asking local bound queries). Our aim is to test whether good recommendations can be generated without the use of conditioning context (which, theoretically, is required).

The third condition, *ADD* uses a simple additive utility model to represent user preferences. No modification of UTPREF is needed, since additive models are a simple, special case of GAI models which have single-attribute factors and empty conditioning sets.

We used a between-group experimental design, with three randomly selected groups of

[1]Utility structure obviously depends on the specific user; however, eliciting factor structure was beyond the scope of the current study. The factors were chosen to reflect a reasonable consensus of possible attribute dependencies based on the domain and an informal survey of potential users.



(a) Rank (b) Qrank

Figure 6.3: Rank and qrank distribution across study participants

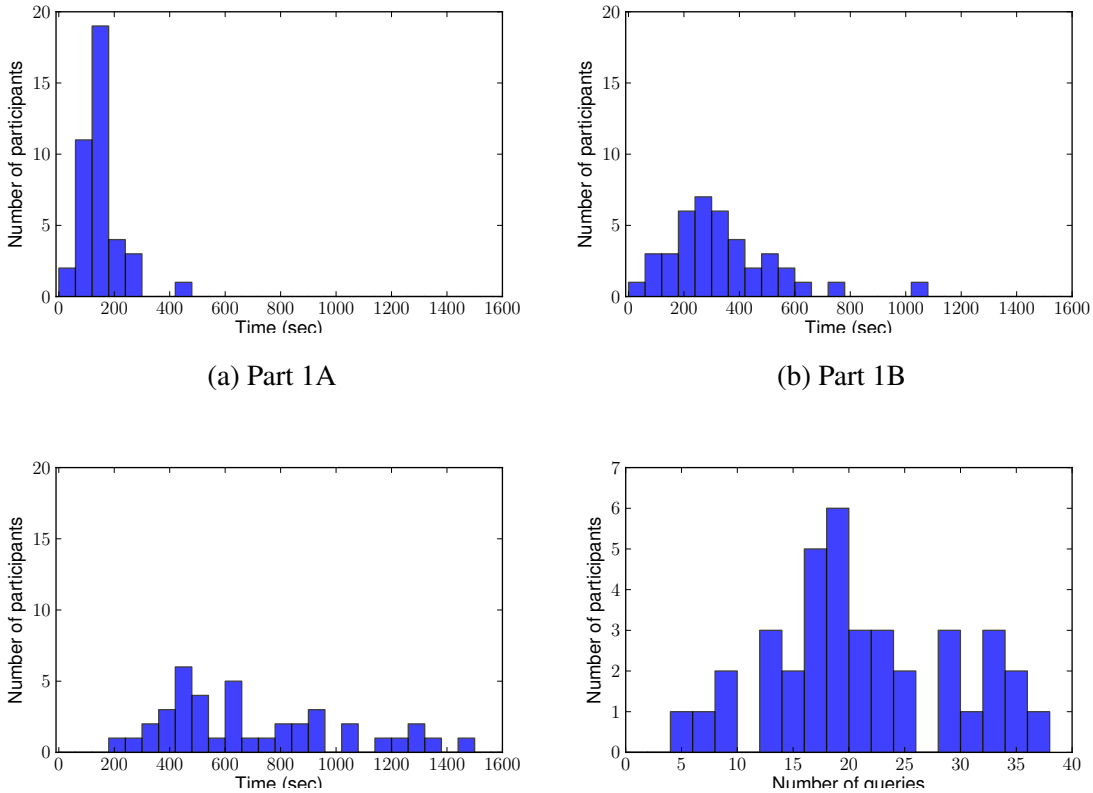
subjects from the 40 participants were assigned to the conditions GAI, GAI-nc and ADD. In addition, half of the users were asked only GCPQ current solution queries, while the other half interacted with the elicitation strategy that employed all other types of queries (ABQ, ACQ, GCQ and LBQ). This gave six subgroups:

	GAI	GAI no context (GAI-nc)	Additive (ADD)
All queries	A	B	C
GCPQ only	D	E	F

All subgroups had 6 participants except subgroups B (7 participants) and C (9 participants).

6.4 User study results

We present the results of the user study in three parts: first we describe overall system performance with respect to recommendation effectiveness, ease of use and user satisfaction; second we discuss query costs (both time and perceived difficulty); and finally we consider performance differences under the different test conditions.



(a) Part 1A

(b) Part 1B

(c) Part 2

(d) Number of queries in Part 1B

Figure 6.4: Completion times for tasks and number of queries. Part 1A, choosing a reference outcome, and sorting all local outcomes for each factor, is the same for all participants; the duration of Part 1B (answering elicitation queries) depends on the number of queries needed to reduce regret to zero. Part 2, rating all 100 apartments in the database, is the same for all participants.

6.4.1 Overall evaluation

User impressions Participants were asked to evaluate a variety of aspects of the UTPREF system after the elicitation process, but before manually rating all apartments. The following questions were rated on a 7-point Likert scale, with scores from 1 ("strongly disagree") to 7 ("strongly agree"); we show average and median responses (with the standard deviation in parentheses):

Some questions were too hard	1.65	2	(0.65)
My answers reflected my true preferences	6.05	6.5	(1.26)
Some questions were confusing	1.98	2	(1.06)
I fully understood the meaning of each question	6.30	6	(0.71)
I answered some questions carelessly	1.98	2	(1.21)
I understood the task I was asked to do	6.75	7	(0.49)
I found this application easy to use	6.35	7	(0.79)
The task took too much time	2.23	2	(1.01)
I am satisfied with the recommended apartment	5.35	5	(1.19)

The average responses to the last four questions confirm that users understood the minimax-regret criterion and were reasonably satisfied with system performance. The majority of participants understood the task well, found UTPREF easy to use, and were satisfied with its recommendation. Note that satisfaction with the recommended apartment is not solely a function of recommendation quality, but is strongly influenced by the available units in the database. We examine recommendation quality next.

Recommendation quality Part 2 of the user session was designed to provide a *quantitative* measure of UTPREF's ability to recommend the "right" product. The major task in Part 2 involved rating all apartments in the database and sorting those rated highest (and those with low minimax regret according to UTPREF), thus providing valuable information about the accu-

racy of the recommended outcome. In the initial rating phase, of 40 participants, 34 assigned the highest rating (2) to the recommended apartment, five rated it 1, and one participant rated it the lowest (0). This suggests that the recommended apartment is almost always a desirable option.

When asked next to sort the final list of 7–12 "very promising" apartments (as described above), users ranked the UTPREF-recommended apartment very highly. The average position (rank) of the recommended apartment was 4.00 (std. dev. 3.32), and the histogram of rankings in Fig. 6.3a shows that 14 users placed the recommended outcome at the top of the sorted list, and majority ranked it among the top three (median 3.0). This too confirms the recommendation accuracy of UTPREF, and is arguably more informative than the coarse 3-point rating of the recommended item.

The position of the recommended apartment in the sorted list does not reveal how from optimal it is in terms of *value*. Asking participants to specify the value difference (in dollars) between the best and worst options in the sorted list lets us roughly quantify the difference between the best apartment in the database and UTPREF's recommendation using a *qrank* score, a quantitative analog of rank (the lower the qrank value, the better). The qrank of an outcome is computed by assuming that apartments in the sorted list are evenly distributed in terms of utility (willingness to pay). For example, if the difference between the best and the worst apartments in a sorted list of 11 apartments is $1000, then the qrank of the top outcome is $0, the worst outcome $1000, and the fifth apartment $400. Fig. 6.3b shows the histogram of the qrank of the recommended apartment across all participants. Only six users have a qrank greater (worse) than $200; equivalently, for 34 of 40 participants, the recommended apartment is within $200 (w.r.t. monthly rent) of the optimal apartment, under qrank assumptions. The average qrank is $100.74 (with std. dev. $151.48); and importantly, the median qrank is $44.95. Thus when the recommended apartment is not optimal it is generally quite close to optimal (within $45 for the majority of users).

Time An important measure of user experience is the duration of the interaction (and the number of queries) required by the recommendation system to find a good product. Fig. 6.4(a-c) shows how long participants took to complete the UTPREF elicitation session (Parts 1A and 1B), and to rate all items in the database (Part 2). Of special significance is the fact that participants spent more time, on average, searching through all apartments in the database (708 seconds) than completing Parts 1A and 1B combined (145 seconds plus 336 seconds) of the elicitation process. This difference is statistically significant ($p = 0.00009$). This holds despite the fact that by the time participants rated the apartments, they had become quite familiar with domain attributes and range of prices, and had time to explore and verify their own preferences by completing the UTPREF elicitation session (Part 1). In addition, increasing the size of the database directly and proportionately impacts the time required to examine all apartments; in fact, we limited our database to 100 apartments because it would be too time-consuming and tedious for users to rate a larger number of options. However, simulation results (Chapter 5, Braziunas and Boutilier (2007)) show that UTPREF can handle larger databases without a significant increase in the number of queries.

Fig. 6.4d shows the histogram of the number of elicitation queries in Part 1B (which excludes choosing the reference outcome, and sorting local outcomes for each factor). Even though the participants had the option to cut the process short after 10 queries, all but two continued until the regret of the recommended apartment dropped to zero. The average number of queries until termination was 21.70 (std. dev. 8.84), and the median was 19.5, illustrating that the majority of users find a near-optimal apartment in under 20 queries. It is also important to note that most of these queries do not require full apartment comparisons and involve only small subsets of attributes.

6.4.2 Query costs

One of the goals of the user study was to estimate the cognitive costs of different query types. We measured the durations and user ratings for the five types of queries described previously:

ACQ, GCQ, GCPQ, ABQ, and LBQ (in addition, local comparison queries were implicitly used in the local outcome sorting task, but were not rated by users). At the end of the elicitation session, participants were asked to rate the query types they encountered—"In terms of difficulty, how would you rate the **type** of question shown?"—on a 7-point Likert scale—ranging from 1 ("extremely difficult") to 7 ("extremely easy"). In addition, the amount of time participants spent responding to each query was recorded. The following table shows average rating and response duration for every query type:

Query type	Avg. rating	Avg. duration (sec)
ACQ	5.28	**13.95**
GCQ	**5.88**	15.58
GCPQ	5.18	14.65
ABQ	5.41	14.86
LBQ	4.23	21.27

A one-way ANOVA that tests whether all the groups are the same (against a hypothesis that they are all different) indicates that the query durations and ratings are significantly different (p=0.00047 and p=0.007, respectively). However, the same test shows no significant differences in ratings or duration among *global* queries (p=0.80 and p=0.56, respectively). In a multiple comparison procedure at the 0.05 level, only LBQs exhibit significant differences from every other query (in durations and ratings).

Both the rating and response time can be used as a proxy for the query's cognitive difficulty. We can see that all global queries have very similar average ratings (between "easy" and "very easy") and response times. Somewhat surprising is the lack of significant differentiation between global bound and comparison queries. In general, users found local bound queries to be the most difficult. From our observations and user comments it is clear that one reason was their novelty and occasional confusion about their meaning; better visual design and explanations might make the LBQs easier to answer. On the other hand, the average LBQ rating of 4.23 still does not suggest serious difficulty (4 corresponds to a query being "not difficult"). As far as we are aware, this is the first evaluation and comparison of query costs in the context of

	ALL	A: all, GAI	B: all, GAI-nc	C: all, ADD	D: GCPQ, GAI	E: GCPQ, GAI-nc	F: GCPQ, ADD
Rank of recommended outcome	4.00	3.33	**3.29**	4.56	4.17	4.33	4.17
Qrank	$101	$106	**$27**	$82	$73	$81	$262
Number of queries	21.70	29.50	27.00	24.56	**13.83**	18.00	15.00
Part 1A duration (sec)	145	168	174	**94**	164	181	110
Part 1B duration (sec)	336	401	500	365	**191**	256	259
Database item rating duration (sec)	708	673	564	828	626	734	787

Table 6.1: Subgroup comparison

preference elicitation systems. This takes a step toward allowing query costs to be traded off against potential query value in future elicitation strategies.

6.4.3 Comparison of study subgroups

The participants were randomly assigned to one of six subgroups A-F (described above), divided along two axes: Query types—all queries vs. global comparison queries (GCPQ) only; and Utility function structure—GAI vs. GAI with no conditioning context (GAI-nc) vs. additive (ADD). Table 6.1 shows the average of the various measures discussed above across the six subgroups.

GAI with and without local context To properly elicit GAI utilities with intersecting factors, local queries have to include certain context attributes. For example (see Fig. 6.2c), a local query about a *house* in *Toronto East* (both building type and area are in Factor 1) also has to specify the reference value of the context attribute *2 bedrooms* (which is in Factor 2), because Factors 1 and 2 intersect. If the context attribute's reference value were different, say *1 bedroom*, user preference for a *house* in *Toronto East* could be very different. To test the importance of local context, local queries for users in subgroups B and E did not include local

context (everything else was the same as in subgroups A and D; for subgroups C and F, this distinction is irrelevant, since additive utility functions do not have intersecting factors). The following table compares key performance metrics:

	All queries		GCPQ only	
	A: GAI	B: GAI-nc	D: GAI	E: GAI-nc
Rank	3.33	**3.29**	**4.17**	4.33
Qrank	$106	**$27**	**$73**	$81
Satisfaction	5.00	**5.86**	5.50	**5.67**

There are no obvious, statistically significant patterns that emerge from the comparison of the GAI and GAI-nc groups. Local queries without context attributes seem to perform no worse than proper, semantically sound queries with local context. There are two plausible explanations for this. First, we believe it is likely that users maintain a consistent context themselves (for instance, some salient outcome like their current or desired apartment) without an explicit specification of it, and automatically assume this context when answering local queries. Confirmation of this hypothesis requires further research. Second, the GAI model used in this study is quite simple, with only two overlapping factors. The importance of proper preferential conditioning generally increases with the complexity of the utility model; thus, differences might emerge in more complex, cognitively demanding choice scenarios.

GAI vs. additive utility models To compare the effectiveness of more flexible GAI models with additive models, we group the GAI and GAI-nc subgroups and compare performance with that of the additive group. The table below compares average Rank, Qrank, and user satisfaction with the recommended apartment for both groups; we show results broken out by query type, and with both query groups combined (as well as p-values in a pairwise comparison):

	Combined			All queries			GCPQ only		
	GAI	ADD	p-	GAI	ADD	p-	GAI	ADD	p-
Subgrp	ABDE	CF	val	AB	C	val	DE	F	val
Rank	**3.76**	4.40	.57	**3.31**	4.56	.38	4.25	**4.17**	.97
Qrank	**$70**	$164	.10	**$63**	$82	.71	**$77**	$262	.07
Satisf.	**5.52**	5.07	.26	**5.46**	4.89	.21	**5.58**	5.33	.68

Although we cannot draw strong, statistically significant conclusions from these results due to the small sample size and high variability, it appears that a more complex GAI utility model leads to better performance with respect to recommendation quality than a simple additive utility model. The only statistically significant advantage is in Qrank (at the 0.1 level). The slight advantage of GAI models is maintained across both types of strategies as well as when both strategy results are combined together. We should bear in mind that the GAI utility factorization used in this study is quite simple, and not tailored to the preference structure of individual participants. In more complex domains, the advantage of a more flexible GAI model could be much more pronounced; but the differences in this study are suggestive enough to warrant further exploration.

All queries vs. GCPQ only queries To explore variation along the query axis, we form the *all queries* group by merging subgroups A,B,C and the *GCPQ only* group by merging subgroups D,E,F. Key performance metrics are shown in the following table (the p-value column shows results from the pairwise comparisons of the first two columns):

	All queries	GCPQ only	p-value
Rank of recommended outcome	**3.82**	4.22	.71
Quantitative rank (qrank)	**$70**	$132	.25
Number of queries	26.68	**15.61**	.000018
Part 1B duration (sec)	418	**235**	.0024

Since global comparison queries concentrate on current solution outcomes, minimax regret tends to decrease very quickly, and on average, the process converges after only 15.61 queries.

The *all queries* strategy uses more queries on average, in large part because it explores a broader utility region by combining both local and global, comparison and bound queries. On the other hand, we suspect that the *all queries* strategy has some advantage with respect to the final recommendation, whose quality is somewhat better in average rank and qrank (though the difference is not statistically significant): the flexibility of its query space allows it to home in on true utility more precisely.

In domains with a relatively small number of attributes (where the user can comfortably compare two arbitrary outcomes), using the GCPQ query is arguably the best option. The strategy reduces minimax regret better than any other strategy (both with simulated and actual users), uses easy-to-understand global comparison queries, and does not take any time for query selection, since it uses the two current MMR solution outcomes. On the other hand, by being able to balance many types of queries, the *all queries* strategy is more flexible and adaptable to a variety of domains with specific query requirements. For example, in domains with large number of attributes, *local* queries become very useful, since they only involve a small subset of attributes.

6.5 Conclusion

6.5.1 Related work

Recommendation systems assist users in the navigation of large product spaces and recommend decisions in the presence of many alternatives. The general research field of recommendation systems is too large to survey here. We focus instead on approaches that construct *explicit* models of user preferences and use those models for recommendations.[1] Some previous decision-

[1]While rarely formulated as developing explicit utility models, some approaches can be so interpreted. For example, in collaborative filtering (Konstan, Miller, Maltz, Herlocker, Gordon, and Riedl, 1997; Breese, Heckerman, and Kadie, 1998; Herlocker, Konstan, Terveen, and Riedl, 2004), matrix factorization methods (Rennie and Srebro, 2005) can be viewed as learning linear user utility models over latent product attributes. *Critiquing systems* (Burke, 2002) facilitate navigation by allowing the user to request changes to a candidate product description in specific dimensions (e.g., increasing one attribute) and returning products that vary in that dimension but

theoretic approaches to preference assessment include work by White et al. (1984); Boutilier et al. (2004c); Salo and Hämäläinen (2001); Toubia et al. (2004). In these models, users answer preference queries incrementally, each response posing constraints on the parameters of a user's utility function. Such models offer stronger guarantees on decision quality, but often at the expense of much more intense data requirements. Furthermore, such models are developed or deployed largely for the case of restrictive *additive* utility models, while strategies for query choice are often ad hoc.

Recently, the *minimax regret* has been proposed as an intuitive criterion for decision making in the presence of utility function uncertainty as well as an effective driver of preference elicitation (Salo and Hämäläinen, 2001; Boutilier et al., 2006). It has been applied to unstructured product models (Wang and Boutilier, 2003), additive models (Boutilier et al., 2004c) and generalized additive models (Boutilier et al., 2006; Braziunas and Boutilier, 2007). In simulation, it has proven to work extremely well. However, until our UTPREF user study, no significant study has explored the effectiveness and intuitive appeal of minimax regret-based elicitation with real users.

Finally, work by Viappiani and Boutilier (2009, 2010); Boutilier, Regan, and Viappiani (2009a,b, 2010) highlights two potential future directions that are very relevant to improving the UTPREF system: additional query types in the form of choice sets (Viappiani and Boutilier, 2009, 2010), and feature (attribute) elicitation (Boutilier et al., 2009a,b, 2010).

6.5.2 Contributions and future work

Explicit user utility modeling with GAI utility functions, semantically sound and user-friendly queries, and minimax regret-based preference elicitation and recommendation are the main features distinguishing UTPREF from other recommendation systems. While effective in simulated experiments, prior to our user study, minimax regret-based elicitation had not been tested

are otherwise similar; Reilly, Zhang, McGinty, Pu, and Smyth (2007); Viappiani et al. (2006) use an explicit user preference model within the critiquing framework.

in realistic domains with real users. Our results from the user study with 40 participants are very encouraging. We demonstrate that minimax regret is an intuitive, comprehensible decision criterion that can be used to drive very effective querying strategies. UTPREF offers high-quality recommendations with minimal user preference revelation. We also show that GAI utility models perform better than simple additive models with respect to several recommendation quality measures. We measure the cognitive costs of different query types, and observe that simple local queries that omit the local context information perform as well as semantically correct local queries.

There are many possible extensions to our current work; some, such as incorporating probabilistic information about user utilities into query strategies, and exploring additional types of queries (Viappiani and Boutilier, 2009, 2010), were discussed in Chapter 5.

A significant current limitation of UTPREF is the assumption that a user's utility function *structure* is known, or can be approximated well by a domain expert. Of course, such an assumption is not always realistic, since preferential dependencies between attributes can vary considerably from user to user. Ideally, an elicitation process would combine assessing preference structure (i.e., determining what sets of attributes are additively independent) with the acquisition of numerical utility parameter values. The process would be optimized so that the user is asked only relevant questions about preferential independence between certain sets of attributes. Solving such a problem is a difficult task. Relevant work includes early decision analysis texts by Fishburn (1970); Keeney and Raiffa (1976), and more recent articles by Chajewska and Koller (2000); Engel and Wellman (2008b, 2010); Baqui (2007); Brafman and Engel (2009, 2010). Utility structure personalization is a natural, although non-trivial, next step in the development of UTPREF.

A related issue is product feature elicitation (Boutilier et al., 2009a), since decision alternatives are not always described in terms of features the user is most comfortable reasoning about. The aim here is to determine the most natural features to describe a product or outcome to al-

low users to most comfortably assess their preferences. For successful practical deployment of recommendation systems we need to address other human-centred issues as well, such as dealing with noisy user responses, verifying the quasilinearity assumption (which implies that users are risk-neutral with respect to item cost), correcting for various ordering and framing biases (Pu et al., 2003), and, in general, making the interaction more natural and intuitive.

The results from the UTPREF user study provide the first assessment of query costs. Incorporating such query costs into elicitation strategies would allow us to investigate the tradeoffs between reducing elicitation effort on the part of the user and improving recommendation quality.

Chapter 7

Conclusion

In this thesis, we presented a decision-theoretic framework for building decision support systems that incrementally learn preferences of individual users over multiattribute outcomes and then provide recommendations based on the acquired preference information. Our framework is based on the well-established models of human preference representation and rational decision making, widely assumed in economics, decision analysis, operations research, AI, and other disciplines (von Neumann and Morgenstern, 1947; Keeney and Raiffa, 1976). By combining such decision-theoretically sound modeling with effective *computational* techniques and certain user-centric considerations, we demonstrate the feasibility and potential of *practical* autonomous preference elicitation and recommendation systems.

Elicitation and recommendation framework

More concretely, in this thesis we focused on decision scenarios in which a user can obtain any outcome from a finite set of available outcomes. The optimal decision is the one that leads to the outcome with the highest utility. The outcome is space is *multiattribute*; each outcome can be viewed as an instantiation of a set of attributes with finite domains. The user has preferences over outcomes that can be represented by a utility function. We assume that user preferences are generalized additively independent, and, therefore, can be represented by a GAI utility

function. GAI utilities provide a flexible representation framework for structured preferences over multiattribute outcomes; they are less restrictive and, therefore, more widely applicable than additive utilities.

In many decision scenarios with large and complex decision spaces (such as making travel plans or choosing an apartment to rent from hundreds of available options), selecting the optimal decision can require a lot of time and effort on the user's behalf. A decision support system can help the decision maker by recommending a decision based on its knowledge of the decision scenario and the acquired knowledge of the user's preferences. While the set of outcomes and available decisions are often the same for many users, user preferences over outcomes can vary substantially from user to user. Since obtaining the user's complete utility function is generally infeasible, the decision support system has to support recommendation with *partial* preference information. If a probabilistic prior about the user's utility function is available, the decision support system evaluates the expected utility of each available option (with expectation taken over all possible utility functions) and recommends the outcome with the highest expected utility. If such prior information is not available, the system represents its uncertainty about user utilities by maintaining a set of *feasible* user utilities and provides recommendations based on the robust minimax regret criterion; that is, it recommends the outcome with the smallest maximum regret (with respect to all adversarial instantiations of feasible utility functions).

When the system's knowledge of the user's preferences is limited, the recommended outcome might be of much lower value than the optimal outcome (with respect to the user's true utility function). By obtaining additional preference information, the system can improve its recommendations. In the elicitation framework considered in this thesis, the decision support system can ask a series of questions about the user's preferences and use the user's responses to reduce its uncertainty about the user's utility function. The interactions are limited to a small set of local and global query types that result in linear constraints on the GAI utility function parameters. The decision support system repeatedly selects a preference query from the set

of available queries, receives a user response, updates its current information about the user's utility function, and continues until termination criteria are met; it then recommends the best outcome with respect to the obtained preference information.

Chapter summaries

Chapter 2 introduces relevant background material from decision and utility theory (such as decision making under certainty and uncertainty, preference structures over multiattribute outcomes, and factored utility representations), and provides a historical and cross-disciplinary overview of previous approaches to preference elicitation. We concentrate on a few key aspects of the problem: what types of queries can be used to extract user preferences, how to represent utility uncertainty, how to make decisions with partial preference information, and how to devise good elicitation strategies. We survey research fields outside computer science (both historical and current) where preference elicitation plays a central role: imprecisely specified multiattribute utility theory (ISMAUT), its extensions to engineering design and configuration problems, conjoint analysis (in marketing), and analytical hierarchy process (in decision analysis). We also provide an overview of recent developments in preference elicitation in the field of AI.

Utility function structure in multiattribute domains is crucial for concise representation and elicitation of preferences. Until recently, additive utility models have enjoyed almost universal popularity both in practice and research. However, there is a tradeoff between simplicity and applicability. *Generalized* additive models are general enough to subsume both simple additive models (by having factors with no shared attributes) and completely unstructured models (by using only one factor altogether). In Chapter 3, we describe the decision-theoretic foundations that support *local* elicitation of GAI utilities and introduce the parameterized representation of GAI utilities that is used throughout the thesis. The first part of the chapter is a detailed introduction to GAI utility models, based on the original work by Fishburn (1967b). The second part deals with semantically sound representation of *local* structure in GAI utilities.

Local structure facilitates not only representation, but also *elicitation* of utility parameters. In the last part of the chapter, we present the set of queries for elicitation of GAI utility parameters that we use in our elicitation framework (in both Bayesian and strict uncertainty settings).

The foundational material in Chapter 3 applies equally to the following chapters, which deal with the Bayesian (Chapter 4) and strict (Chapter 5) representation of uncertainty over possible GAI utility functions. In each case, two issues are addressed: 1) how to make good decisions when full utility information is not available; and, 2) what is the best elicitation strategy when further preference information can be obtained. When uncertainty over utilities is quantified probabilistically, Bayesian principles can be applied to both decision making and elicitation of user preferences. In Chapter 4, we propose a mixture-of-uniforms probability model to specify uncertainty over the local parameters of the GAI utility function and use it to perform effective elicitation driven by a myopic EVOI strategy. We conclude the chapter with experimental results on a few large problems with simulated user utilities.

Chapter 5 deals with GAI utility elicitation under strict uncertainty. We present tractable algorithms for utility elicitation for configuration (using mixed integer programs) and database (using intelligent search techniques) problems. The minimax regret criterion is used for both decision making and elicitation under utility function uncertainty. We finish the chapter by comparing the performance of various elicitation strategies on several configuration and database problems (with simulated users).

Chapter 6 focuses on evaluation of the minimax regret-based approach to preference elicitation and recommendation. In the first part of the chapter, we present the UTPREF recommendation system that searches multiattribute product databases using the minimax regret criterion. In the second part, we report on a study involving 40 users interacting with the UTPREF system, configured to help users find rental accommodation in Toronto. The study is designed to test the effectiveness of regret-based elicitation, evaluate user comprehension and acceptance of minimax regret, and assess the relative difficulty of different query types.

The thesis chapters are based on the following publications:

- *Local Utility Elicitation in GAI Models* (Braziunas and Boutilier, 2005): Chapters 3 and 4;
- *Minimax Regret-based Elicitation of Generalized Additive Utilities* (Braziunas and Boutilier, 2007): Chapters 5 and 6;
- *Elicitation of Factored Utilities* (Braziunas and Boutilier, 2008): Chapter 2;
- *Assessing Regret-based Preference Elicitation with the UTPREF Recommendation System* (Braziunas and Boutilier, 2010): Chapter 6.

Contributions

Here, we summarize the main contributions of this thesis. In the following section, we discuss the limitations and suggest directions for future work.

- **GAI utility function parameterization in terms of global and local parameters**

  In Chapter 3, we present a locally parameterized GAI model that, together with appropriate types of queries introduced in Section 3.3, can be used for effective and decision-theoretically sound elicitation of GAI utilities.

  To design proper elicitation techniques for GAI models, we first have to solve the problem of subutility function semantics. Unlike additive models, GAI models require much more care in calibration because of the possible overlap of factors (sharing of attributes). If factors overlap, there are infinitely many valid decompositions of the same utility function in which the subutility functions vary considerably (i.e., not simply through some positive affine transformation). It is quite possible that the apparent "local preferences" for factor instantiations can be reversed in two different valid representations. Our identification of this problem is one of the main contributions of Chapter 3.[1]

[1]The GAI elicitation technique proposed by Gonzales and Perny (2004) uses global outcome queries and implicitly acknowledges the same issue.

Our solution to this problem rests on using Fishburn's original canonical representation of subutility functions (Fishburn, 1967b) as a basis for decision-theoretically sound representation and elicitation of GAI utilities. We introduce a *parameterized* representation of GAI utilities that preserves the local structure of additive models, analyze its properties, and provide a graphical search algorithm for computing the GAI structure parameters. By taking into account the *conditioning sets* of attributes that shield the influence of other attribute values on local preferences over factor instantiations, we generalize semantically sound local value functions of additive models to GAI models. As in additive models, the LVFs calibrate local preferences relative to the best and worst factor suboutcomes, assuming fixed values of the attributes in the conditioning set. The LVFs are local, because they involve only attributes in single factors and their (usually small) conditioning sets.

Using our representation, GAI models can be elicited by using both local queries about preferences over small subsets of attributes and global queries for calibration across utility factors. In Section 3.3, we present several types of semantically sound local and global queries that are both easy for users to understand and respond, and result in linear constraints on GAI utility parameters. Such queries form the basic interaction units in our elicitation framework.

- **Bayesian elicitation of GAI utilities**

  Assuming that the system has prior probabilistic information about the GAI local value function parameters, we present an algorithm for Bayesian utility elicitation and decision making with GAI utilities (previous work assumed flat utility representations). In particular, we show that if the priors over local parameters are specified using mixtures of uniforms, then the best myopic local bound query on a specific LVF parameter can be computed analytically. This is due to the fact that mixtures of uniforms are closed under updates resulting from bound queries, which makes it possible to maintain an exact density over utility parameters throughout the elicitation process. We use this result to

develop a tractable procedure for myopically optimal preference elicitation. Experimental results with simulated user utilities confirm the benefits of our approach and show that it can potentially support interactive real-time elicitation.

- **Minimax-regret based elicitation of GAI utilities**

  We extend the previous work of Boutilier et al. (2001, 2003b, 2005, 2006) on applying minimax regret to elicitation of GAI utility models in several ways. First, we show how to incorporate *semantically sound* local queries into GAI preference elicitation framework, and how to compute minimax regret in configuration problems using the parametric representation of GAI models. Second, we also consider database domains, and apply minimax search pruning techniques to speed-up regret computation in databases domains. Third, for elicitation, we extend the current solution heuristic idea to derive scores for *all* types of queries described in Sec 3.3; specifically, the incorporation of *comparison* queries into the framework is of great practical significance, since such queries are very natural for users to understand and respond to (see Chapter 6).

- **Evaluation of the minimax-regret based elicitation framework with real users**

  The minimax regret-based recommendation system UTPREF is the first system that: a) explicitly uses GAI utility functions for internal preference representation; b) employs semantically sound and user-friendly preference queries; and, c) uses the minimax regret criterion for preference elicitation and recommendation. While effective in simulated experiments, prior to our user study, minimax regret-based elicitation had not been tested in realistic domains with real users. We conducted a user study with 40 participants and obtained very encouraging results. We demonstrate that minimax regret is an intuitive, comprehensible decision criterion that can be used to drive effective querying strategies. UTPREF offers high-quality recommendations with minimal user preference revelation. We also show that GAI utility models perform better than simple additive models with

respect to several recommendation quality measures. We measure the cognitive costs of different query types, and observe that simple local queries that omit the local context information perform as well as semantically correct local queries.

Future work

Our proposed framework for decision-theoretic preference elicitation is just one of the many steps towards the goal of building successful autonomous recommendation systems. Many aspects of the framework can be improved and extended. Here, we list some of its limitations and promising future research directions.

- **GAI structure assessment** One assumption that we make in this thesis is that a user's utility function *structure* is known, or can be approximated well by a domain expert. Of course, such an assumption is not always realistic, since preferential dependencies between attributes can vary considerably from user to user. Ideally, an elicitation process would combine assessing preference structure (i.e., determining what sets of attributes are additively independent) with the acquisition of numerical utility parameter values. The process would be optimized so that the user is asked only relevant questions about preferential independence between certain sets of attributes.

  Solving such a problem is a difficult task. Some previous work of relevance include early decision analysis texts by Fishburn (1970); Keeney and Raiffa (1976), and more recent work by Chajewska and Koller (2000); Engel and Wellman (2008b, 2010); Baqui (2007); Brafman and Engel (2009, 2010). Utility structure personalization is a natural, although non-trivial, next step in the development of UTPREF.

- **Extending the elicitation framework**

  The elicitation framework described in this thesis is quite rigid: users interact with the decision support system by providing responses to a limited set of queries. We assume that users can clearly articulate their preferences and provide meaningful, accurate and

consistent responses. Most users are not experts and therefore require preliminary training and easy-to-understand interfaces. Real case studies often provide evidence of inconsistent responses, errors, and various forms of biases (Simon, 1955; Tversky and Kahneman, 1974; Camerer et al., 2003; Pu et al., 2003).

Successful implementation of decision support systems requires consideration of both descriptive and prescriptive approaches to the preference elicitation problem. In addition to the sound decision-theoretic framework, we also need to address human-centred issues, such as framing and ordering effects, sensitivity analysis and robustness, and the reliability and acceptability of different modes of interaction (Pu et al., 2003). Such empirical validation is only obtainable with further user studies, which can make our interaction models more natural and understandable for end users, as well as more effective in providing good recommendations.

The queries that we use in our framework are well grounded in terms of decision-theoretic semantics and simple enough to be used in actual interactive applications with human subjects. However, there is a variety of additional preference query types and modes of interaction that can be effective in eliciting human preferences. *Conversational recommender systems* provide mixed-initiative interactions, with both the system and the user working together to optimize the exploration of large outcome spaces. In such systems, user feedback can be obtained as direct responses to preference queries, or a critique of displayed options, either by choosing one option, or reconfiguring some given options (Burke, 2002; Pu and Faltings, 2004; Viappiani et al., 2006; Reilly et al., 2007; Viappiani and Boutilier, 2009, 2010). For each new query type, we have to provide a user interface, incorporate it into our modeling framework, and devise good heuristic scoring functions for use in mixed query strategies.

A related issue is product feature elicitation (Boutilier et al., 2009a), since decision alternatives are not always described in terms of features the user is most comfortable reasoning about. The aim here is to determine the most natural features to describe a

product or outcome to allow users to most comfortably assess their preferences. For successful practical deployment of recommendation systems we need to address other human-centred issues as well, such as dealing with noisy user responses, verifying the quasilinearity assumption (which implies that users are risk-neutral with respect to item cost), correcting for various ordering and framing biases, and, in general, making the interaction more natural and intuitive.

- **Improvements to Bayesian elicitation**

  There are many ways to represent probabilistic uncertainty over utilities. In this thesis, we only explored the setting in which the system has a mixture-of-uniforms prior over LVF parameters. Investigating other forms of prior distributions (over possibly different utility parameter sets) remains an open research direction. A related problem is acquisition of prior knowledge. Priors could be provided by experts, or learned from data (Chajewska et al., 1998). Although public utility databases are very scarce, their availability is increasing (Portabella Clotet and Rajman, 2006; Braziunas and Boutilier, 2010). Recent advances in collaborative filtering on large datasets of product ratings by hundreds of thousands of users can be leveraged to provide informative priors for groups of users with similar interests (Breese et al., 1998; Adomavicius and Tuzhilin, 2005; Salakhutdinov and Mnih, 2007).

  To improve the Bayesian elicitation procedure, we need to incorporate more query types (we only discussed local bound queries that, for a given query parameter, allow us to compute the best query bound analytically with mixture-of-uniforms uncertainty representation). Comparison queries present serious challenges when uncertainty is represented by a parametric probability model since responses to comparison queries impose "diagonal" (rather than axis-parallel) constraints on posterior distributions. This requires sampling and refitting for belief state maintenance and also complicates query optimization. However, incorporation of more types of queries is vital for practical systems, since users are likely to be more comfortable with simpler binary comparison queries

than bound queries.

- **Combined Bayesian-MMR approach** Minimax regret is a robust decision criterion, providing decision quality guarantees for any possible realization of the user's utility function. In some domains, it is desirable to use the minimax regret criterion for final recommendation, even though prior probabilistic information about user utilities is available. We could take advantage of probabilistic information about user utilities in optimizing elicitation strategy, but still use the robust MMR decision criterion for the final decision (Wang and Boutilier, 2003). Combining the two approaches remains a promising future research area.

- **Sequentially-optimal elicitation** For computational reasons, we only explored myopic query strategies that are not *sequentially* optimal because they do not consider the impact of future queries when computing the value of the decision support systems information state (represented by a belief state in the Bayesian elicitation setting, and by a set of feasible utility functions in the strict uncertainty setting). Solving for the sequentially optimal query policy is equivalent to solving a POMDP with a continuous state space, which is a very hard problem (Boutilier, 2002). Several approaches for solving the preference POMDP have been proposed, including value function approximation (Boutilier, 2002) and searching for good finite controller policies (Braziunas and Boutilier, 2004). However, more research is needed to scale these solutions to realistic problems. Another direction would be for the system to perform more than one-step lookahead *online* when evaluating queries (if the system has enough time between queries). With a deep lookahead horizon, the value of a query would approach its sequentially optimal value.

Appendix A

ISMAUT

From section 2.3.1, J is the set of prior pairwise preferences between alternatives in A, $R(\mathbf{U})$ is the derived set of pairwise preferences between alternatives given the feasible utility set $\mathbf{U}$, and $ND(\mathbf{U})$ is the set of nondominated alternatives. The set $ND(\mathbf{U})$ can be straightforwardly computed from $R(\mathbf{U})$. The central computational task is therefore to compute $R(\mathbf{U})$ from $\mathbf{U}$. Recall that $(\mathbf{x}, \mathbf{y}) \in R(\mathbf{U})$ if and only if $\mathbf{w} \cdot [\mathbf{v}(\mathbf{x}) - \mathbf{v}(\mathbf{y})] \geq 0$, for all $\langle \mathbf{w}, v_1, \ldots, v_n \rangle \in \mathbf{U}$. This amounts to verifying that

$$\min_{\langle \mathbf{w}, v_1, \ldots, v_n \rangle \in \mathbf{U}} \mathbf{w} \cdot [\mathbf{v}(\mathbf{x}) - \mathbf{v}(\mathbf{y})] \geq 0.$$

If $J = \emptyset$, then there is no prior information about pairwise preferences between alternatives, and no coupling between the weights and value functions. In this case, researched by White et al. (1983), Eq. 2.33 becomes

$$\min_{\mathbf{w} \in W} \left(\sum_{i=1}^{N} w_i \min_{v_i \in V_i} [v_i(x_i) - v_i(y_i)] \right) \geq 0. \tag{A.1}$$

This problem can be solved by $N + 1$ linear programs.

Adding constraints to the set J complicates the solution, because the weight and value function constraints get tied together by comparisons of global alternatives. Anandalingam and White (1993) propose a general penalty function method for determining membership in $R(\mathbf{U})$. However, since such methods might suffer from slow convergence and ill-conditioning

(Bazaraa and Shetty, 1979), two special cases of approximating $R(\mathbf{U})$ are analyzed by White et al. (1984), and Anandalingam and White (1993).

Let $\mathbf{U}^{min}$ be a set of tuples $\langle \mathbf{w}, v_1, \ldots, v_n \rangle$ such that $\mathbf{w} \in W$, $v_i \in V_i$, and for all $(\mathbf{x}^j, \mathbf{y}^j) \in J$,

$$\sum_{i=1}^{N} w_i \min_{v_i \in V_i} [v_i(x_i^j) - v_i(y_i^j)] \geq 0. \tag{A.2}$$

Similarly, let $\mathbf{U}^{max}$ be a set of tuples $\langle \mathbf{w}, v_1, \ldots, v_n \rangle$ such that $\mathbf{w} \in W$, $v_i \in V_i$, and for all $(\mathbf{x}^j, \mathbf{y}^j) \in J$,

$$\sum_{i=1}^{N} w_i \max_{v_i \in V_i} [v_i(x_i^j) - v_i(y_i^j)] \geq 0. \tag{A.3}$$

Inequalities A.2 and A.3 impose linear constraints on weights. Therefore, solving for $R(\mathbf{U}^{min})$ and $R(\mathbf{U}^{max})$ is straightforward, since this case is similar to the $J = \emptyset$ case described above.

How can $\mathbf{U}^{min}$ and $\mathbf{U}^{max}$ be used to approximate $R(\mathbf{U})$? From Eq. A.2 and A.3, we note that $\mathbf{U}^{min} \subseteq \mathbf{U} \subseteq \mathbf{U}^{max}$, and therefore, $R(\mathbf{U}^{max}) \subseteq R(\mathbf{U}) \subseteq R(\mathbf{U}^{min})$. Without computing $R(\mathbf{U})$, we can infer two facts:

$$(\mathbf{x}, \mathbf{y}) \in R(\mathbf{U}^{max}) \implies (\mathbf{x}, \mathbf{y}) \in R(\mathbf{U}),$$
$$(\mathbf{x}, \mathbf{y}) \notin R(\mathbf{U}^{min}) \implies (\mathbf{x}, \mathbf{y}) \notin R(\mathbf{U}).$$

Therefore, it is only when $(\mathbf{x}, \mathbf{y}) \notin R(\mathbf{U}^{max})$ or $(\mathbf{x}, \mathbf{y}) \in R(\mathbf{U}^{min})$ that membership in $R(\mathbf{U})$ needs to be determined directly.

Appendix B

GAI representation theorem

This derivation is part of the proof of the GAI representation theorem (Section 3.1.2), based on the original work by Fishburn (1967b).

The two marginal distributions P_j and Q_j are:

$$P_j = \alpha \mathbf{x}_j + \sum_{\substack{k \geq 2, \\ k \text{ even}}} \sum_{S \in R_k} \alpha \mathbf{x}_j [\cap_{s \in S} I_s],$$

$$Q_j = \sum_{\substack{k \geq 1, \\ k \text{ odd}}} \sum_{S \in R_k} \alpha \mathbf{x}_j [\cap_{s \in S} I_s],$$

where $\alpha = \frac{1}{2^{M-1}}$, R is the power set of factor indices $\{1, \ldots, M\}$, and $R_k \subseteq R$ (for $k = 0, \ldots, M$) is the set of subsets in R whose size is k. Consequently, the size of R_k is $\binom{M}{k}$.

We need to show that $P_j = Q_j$:

$$
\begin{aligned}
P_j &= \alpha\mathbf{x}_j + \sum_{\substack{k\geq 2,\\ k \text{ even}}} \sum_{S\in R_k} \alpha\mathbf{x}_j[\cap_{s\in S} I_s] \\
&= \alpha\mathbf{x}_j + \sum_{\substack{k\geq 2,\\ k \text{ even}}} \left(\sum_{\substack{S\in R_k,\\ j\in S}} \alpha\mathbf{x}_j[\cap_{s\in S} I_s] + \sum_{\substack{S\in R_k,\\ j\notin S}} \alpha\mathbf{x}_j[\cap_{s\in S} I_s] \right) \\
&= \alpha\mathbf{x}_j + \sum_{\substack{k\geq 2,\\ k \text{ even}}} \left(\sum_{\substack{S\in R_{k-1},\\ j\notin S}} \alpha\mathbf{x}_j[\cap_{s\in S} I_s] + \sum_{\substack{S\in R_k,\\ j\notin S}} \alpha\mathbf{x}_j[\cap_{s\in S} I_s] \right) \\
&= \alpha\mathbf{x}_j + \sum_{k=1}^{M} \sum_{\substack{S\in R_k,\\ j\notin S}} \alpha\mathbf{x}_j[\cap_{s\in S} I_s] \\
&= \alpha\mathbf{x}_j + \sum_{\substack{S\in R_1,\\ j\notin S}} \alpha\mathbf{x}_j[I_s] + \sum_{\substack{k\geq 3,\\ k \text{ odd}}} \left(\sum_{\substack{S\in R_{k-1},\\ j\notin S}} \alpha\mathbf{x}_j[\cap_{s\in S} I_s] + \sum_{\substack{S\in R_k,\\ j\notin S}} \alpha\mathbf{x}_j[\cap_{s\in S} I_s] \right) \\
&= \sum_{\substack{S\in R_1,\\ j\in S}} \alpha\mathbf{x}_j[I_s] + \sum_{\substack{S\in R_1,\\ j\notin S}} \alpha\mathbf{x}_j[I_s] + \sum_{\substack{k\geq 3,\\ k \text{ odd}}} \left(\sum_{\substack{S\in R_k,\\ j\in S}} \alpha\mathbf{x}_j[\cap_{s\in S} I_s] + \sum_{\substack{S\in R_k,\\ j\notin S}} \alpha\mathbf{x}_j[\cap_{s\in S} I_s] \right) \\
&= \sum_{\substack{k\geq 1,\\ k \text{ odd}}} \left(\sum_{\substack{S\in R_k,\\ j\in S}} \alpha\mathbf{x}_j[\cap_{s\in S} I_s] + \sum_{\substack{S\in R_k,\\ j\notin S}} \alpha\mathbf{x}_j[\cap_{s\in S} I_s] \right) \\
&= \sum_{\substack{k\geq 1,\\ k \text{ odd}}} \sum_{S\in R_k} \alpha\mathbf{x}_j[\cap_{s\in S} I_s] \\
&= Q_j.\square
\end{aligned}
$$

Appendix C

Domains

C.1 Car rental domain

The car-rental problem is modeled with 26 variables that specify various attributes of a car relevant to typical rental decisions (Boutilier et al., 2003b, 2005, 2006). Variable domains range from 2 to 9 values, resulting in 61,917,364,224 possible configurations. The GAI model consists of 13 local factors, each defined on at most five variables; the model has 378 utility parameters.

Attributes

1. *Car class*: `economy`, `compact`, `medium`, `full`, `premium`, `luxury`, `van`, `suv`, `convertible`
2. *Number of cylinders*: `4`, `6`, `8`
3. *Seat capacity*: `4`, `5`, `6`, `7`
4. *Traction type*: `front`, `rear`, `4wd`
5. *Cruise control*: `no`, `yes`
6. *Sound system*: `radio`, `radio_cassette`, `radio_cassette_cd`
7. *Storage capacity*: `2`, `3`, `4`
8. *Child safety lock*: `no`, `yes`
9. *Power steering*: `no`, `yes`

10. *Airbags*: `no`, `driver`, `both`
11. *Air conditioning*: `no`, `yes`
12. *Transmission*: `manual`, `auto`
13. *ABS*: `no`, `yes`
14. *Car origin*: `import`, `domestic`
15. *Car manufacturer*: `kia`, `hyundai`, `chevy`, `gm`, `chrysler`, `ford`
16. *Rental agency*: `national`, `avis`, `enterprise`, `thrifty`, `dollar`, `budget`
17. *Locks*: `key`, `power`, `keyless`
18. *Remote locks*: `no`, `yes`
19. *Power windows*: `no`, `yes`
20. *Sunroof*: `no`, `yes`
21. *Tinted windows*: `no`, `yes`
22. *Climate control*: `no`, `yes`
23. *Power driver seat*: `no`, `yes`
24. *Steering wheel tilt*: `no`, `yes`
25. *Delay wipers*: `no`, `yes`
26. *Power mirrors*: `no`, `yes`

Factors

- $F_1 = \{$*Car class*, *Child safety lock*$\}$
- $F_2 = \{$*Number of cylinders*, *Seat capacity*$\}$
- $F_3 = \{$*Car class*, *Traction type*$\}$
- $F_4 = \{$*Car class*, *Storage capacity*$\}$
- $F_5 = \{$*Car class*, *Number of cylinders*, *Cruise control*, *Sound system*$\}$
- $F_6 = \{$*Air conditioning*, *Power windows*$\}$
- $F_7 = \{$*Car origin*, *Locks*, *Remote locks*, *Power mirrors*$\}$

- $F_8 = \{$*Car manufacturer*, *Rental agency*$\}$
- $F_9 = \{$*Car class*, *Transmission*$\}$
- $F_{10} = \{$*Remote locks*, *Power windows*, *Sunroof*, *Tinted windows*, *Steering wheel tilt*$\}$
- $F_{11} = \{$*Power steering*, *Airbags*$\}$
- $F_{12} = \{$*Transmission*, *ABS*$\}$
- $F_{13} = \{$*Climate control*, *Power driver seat*, *Delay wipers*$\}$

C.2 Apartment rental domain

The apartment rental problem comprises a database of 200 apartments, described by ten attributes (including price), each having between two and four domain values (Braziunas and Boutilier, 2010). The GAI model has eight factors.

Attributes

1. *Area*: `TorontoCentral`, `TorontoEast`, `TorontoWest`, `Scarborough`
2. *Building type*: `house`, `apartment`, `basement`
3. *No. of bedrooms*: `1bedroom`, `2bedrooms`, `3bedrooms`
4. *Furniture*: `unfurnished`, `furnished`
5. *Laundry*: `available`, `notavailable`
6. *Parking*: `available`, `notavailable`
7. *Dishwasher*: `yes`, `no`
8. *Storage*: `yes`, `no`
9. *Air conditioning*: `yes`, `no`
10. *Price*: $\$600 - \$1800$

Factors

- $F_1 = \{\textit{Area}, \textit{Building type}\}$
- $F_2 = \{\textit{Building type}, \textit{No. of bedrooms}\}$
- $F_3 = \{\textit{Furniture}\}$
- $F_4 = \{\textit{Laundry}\}$
- $F_5 = \{\textit{Parking}\}$
- $F_6 = \{\textit{Dishwasher}\}$
- $F_7 = \{\textit{Storage}\}$
- $F_8 = \{\textit{Air conditioning}\}$

Appendix D

Minimax search with alpha-beta pruning

```
def minimax(getValueFn, N, pruning='alpha-beta', MINstart=0):
    """Perform 2-ply minimax search with pruning

    getValueFn(i,j) is the function that returns terminal value
    N is the number of rows and columns (MIN choices and MAX choices)
    pruning: none, alpha, beta, alpha-beta
    MINstart is the initial MIN choice to consider
    """

    # betai will eventually be the optimal MIN choice
    # beta keeps track of the MIN node's upper bound
    betai, beta = None, Infinity

    # rows.keys() are row indices (MIN choices)
    # rows.values() are corresponding alphas (max known element in a row)
    # initialize all alphas to -Infinity
    rows = {}
    for i in range(0, N):
```

```
19          rows[i] = -Infinity
20
21      # Js[i] is a list of unexplored MAX choices for every row (MIN choice) i
22      # initially, Js[i] is a list of numbers from 0 to N-1 for every i
23      Js = [range(0, N) for i in range(0, N)]
24
25      # start is the first MIN choice (row) to try (default: 0)
26      curri = MINstart
27
28      # in the following main loop,
29      # curri will be current MIN choice (row) being explored
30      while len(rows) > 0:
31          alpha = rows[curri]
32          assert(alpha < beta)
33          # remove current row from search frontier
34          del rows[curri]
35
36          # maxj is what MAX would play if MIN chose curri
37          # MAX's choices are limited to Js[curri]
38          # alpha pruning happens inside maxvalue based on beta
39          if pruning in ['alpha', 'alpha-beta'] :
40              maxj, maxval = maxvalue(getValueFn, curri, Js[curri], beta=beta)
41          else:
42              # 'none', 'beta' (no alpha ppruning)
43              maxj, maxval = maxvalue(getValueFn, curri, Js[curri], beta=Infinity)
44
45          # maxval is only the largest value among Js[curri]
46          # if alpha (computed earlier) is larger,
47          # then max element in row curri is alpha
```

```
48          alpha = max(alpha, maxval)
49
50          # update beta
51          if alpha < beta:
52              betai, beta = curri, alpha
53
54
55          # NEXT MIN CHOICE
56
57          # choose any available MIN choice (row) if no beta-pruning
58          # return (continue) to the beginning of the loop
59          if pruning in ['none', 'alpha'] and len(rows) > 0:
60              curri = rows.keys()[0]
61              continue
62
63          # beta pruning happens below
64          # strategy: choose MIN row that minimizes current MAX choice maxj
65          # assumption: maxj is a good MAX choice for many MIN choices
66          # (not just curri)
67
68          # go through all available MIN choices (rows.keys()) and
69          # find the one with minimum value (with respect to maxj)
70          mini, minval = None, Infinity
71          for i in rows.keys():
72              alpha = rows[i]
73              # beta pruning:
74              # if row i max known element is already greater than beta,
75              # row i will never be chosen, so we remove it
76              # and go to next row
```

```
            if alpha >= beta:
                del rows[i]
            else:
                v = getValueFn(i, maxj)
                # update alpha
                if v > alpha:
                    alpha = rows[i] = v

                # beta pruning after computing v
                if alpha >= beta:
                    del rows[i]
                else:
                    # remove MAX choice maxj from further consideration
                    # because its value was already computed
                    Js[i].remove(maxj)
                    # update minimum
                    if v < minval:
                        mini, minval = i, v

        # next MIN choice is the one that minimizes MAX's choice maxj
        curri = mini
    return betai, beta
```

maxvalue subroutine

```
def maxvalue(getValueFn, i, Js, beta=Infinity):
    """Return largest value among Js (with alpha pruning)

    getValueFn(i,j) is the function that returns terminal value
    i is the MINchoice (row)
```

```
    Js are the poassible MAXchoices (columns)

    Function returns the first value among Js
    that is larger than beta (alpha pruning) or,
    if all Js are less than beta, than the largest j
    """

    # highest value known so far
    maxj, maxval = None, -Infinity
    for j in Js:
        v = getValueFn(i,j)
        # alpha pruning:
        # row i's maximum value is at least v,
        # and if v is greater than beta
        # the row will never be chosen by the MIN agent.
        # Therefore, there's no need to check remaining entries
        if v >= beta:
            return j, v

        # else, update maxval
        elif v > maxval:
            maxj, maxval = j, v
    return maxj, maxval
```

Bibliography

Ali Abbas. Entropy methods for adaptive utility elicitation. *IEEE Transactions on Systems, Science and Cybernetics*, 34(2):169–178, 2004. 49, 50, 168

Gediminas Adomavicius and Alexander Tuzhilin. Towards the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6), 2005. 264

Selim G. Akl and Monroe M. Newborn. The principle continuation and the killer heuristic. In *Proceedings of the ACM Annual Conference*, pages 466–473, 1977. 199

G. Anandalingam and Chelsea C. White. A penalty function approach to alternative pairwise comparisons in ISMAUT. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(1): 330–333, 1993. 54, 136, 266, 267

Fahiem Bacchus and Adam Grove. Graphical models for preference and utility. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pages 3–10, Montreal, 1995. 4, 34, 37, 75, 76, 107, 123

Fahiem Bacchus and Adam Grove. Utility independence in qualitative decision theory. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning*, pages 542–552, Cambridge, 1996. 28

Abeera Farzana Al Baqui. Value-focused GAI network structure elicitation given a domain ontology. Master's thesis, University of British Columbia, 2007. 125, 253, 262

Mokhtar S. Bazaraa and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. Wiley, New York, 1979. 267

J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962. 191

Umberto Bertele and Francesco Brioschi. *Nonserial Dynamic Programming*. Academic Press, Orlando, 1972. 137

Jim Blythe. Visual exploration and incremental utility elicitation. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02)*, pages 526–532, Edmonton, 2002. 136

Board on Manufacturing and Engineering Design. *Theoretical Foundations for Decision Making in Engineering Design*. National Academies Press, 2001. 57

Edwin V. Bonilla, Shengbo Guo, and Scott Sanner. Gaussian process preference elicitation. In *Advances in Neural Information Processing Systems 23 (NIPS-10)*, Vancouver, 2010. 167, 168

Craig Boutilier. Toward a logic for qualitative decision theory. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, pages 75–86, Bonn, 1994. 28

Craig Boutilier. A POMDP formulation of preference elicitation problems. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02)*, pages 239–246, Edmonton, 2002. 10, 37, 41, 48, 49, 53, 69, 70, 132, 141, 151, 152, 167, 168, 169, 170, 265

Craig Boutilier. On the foundations of *expected* expected utility. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 285–290, Acapulco, 2003. 48, 49, 131, 132

Craig Boutilier, Ronen Brafman, Chris Geib, and David Poole. A constraint-based approach to preference elicitation and decision making. In *AAAI Spring Symposium on Qualitative Decision Theory*, Stanford, 1997. 58

Craig Boutilier, Ronen I. Brafman, Holger H. Hoos, and David Poole. Reasoning with conditional ceteris paribus preference statements. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 71–80, Stockholm, 1999. 28, 89

Craig Boutilier, Fahiem Bacchus, and Ronen I. Brafman. UCP-Networks: A directed graphical representation of conditional utilities. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pages 56–64, Seattle, 2001. 4, 6, 11, 29, 31, 34, 37, 45, 46, 89, 124, 176, 179, 227, 261

Craig Boutilier, Rajarshi Das, Jeffrey O. Kephart, Gerald Tesauro, and William E. Walsh. Cooperative negotiation in autonomic systems using incremental utility elicitation. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI-03)*, pages 89–97, Acapulco, 2003a. 46, 50, 70, 71

Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Constraint-based optimization with the minimax decision criterion. In *Ninth International Conference on Principles and Practice of Constraint Programming*, pages 168–182, Kinsale, Ireland, 2003b. 6, 10, 34, 40, 41, 46, 50, 70, 124, 163, 179, 227, 261, 270

Craig Boutilier, Ronen Brafman, Carmel Domshlak, Holger Hoos, and David Poole. Preference-based constrained optimization with cp-nets. *Computational Intelligence*, 20 (2):137–157, 2004a. 29

Craig Boutilier, Ronen Brafman, Carmel Domshlak, Holger Hoos, and David Poole. Cp-networks: A tool for representing and reasoning with conditional *Ceteris Paribus* preference statements. *Journal of Artificial Intelligence Research*, 21:135–191, 2004b. 89, 90

Craig Boutilier, Tuomas Sandholm, and Rob Shields. Eliciting bid taker non-price preferences in (combinatorial) auctions. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, pages 204–211, San Jose, CA, 2004c. 12, 46, 50, 70, 176, 205, 227, 252

Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Regret-based utility elicitation in constraint-based decision problems. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 929–934, Edinburgh, 2005. 6, 34, 41, 46, 50, 70, 71, 124, 163, 179, 227, 261, 270

Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artifical Intelligence*, 170(8–9):686–713, 2006. 4, 11, 12, 124, 138, 163, 176, 179, 191, 204, 205, 210, 227, 252, 261, 270

Craig Boutilier, Kevin Regan, and Paolo Viappiani. Online feature elicitation in interactive optimization. In *Proceedings of the Twenty-sixth International Conference on Machine Learning (ICML-09)*, pages 73–80, Montreal, 2009a. 252, 253, 263

Craig Boutilier, Kevin Regan, and Paolo Viappiani. Preference elicitation with subjective features. In *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys09)*, pages 341–344, New York, 2009b. 252

Craig Boutilier, Kevin Regan, and Paolo Viappiani. Simultaneous elicitation of preference features and utility. In *Proceedings of the Twenty-fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, pages 1160–1167, Atlanta, 2010. 252

Ronen Brafman and Carmel Domshlak. Introducing variable importance tradeoffs into CP-nets. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02)*, pages 69–76, Edmonton, 2002. 29

Ronen Brafman and Yagil Engel. Directional decomposition of multiattribute utility functions. In *Proceedings of the First Conference on Algorithmic Decision Theory (ADT-09)*, Venice, 2009. 123, 126, 253, 262

Ronen Brafman and Yagil Engel. Decomposed utility functions and graphical models for reasoning about preferences. In *Proceedings of the Twenty-fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, Atlanta, Georgia, 2010. 123, 126, 253, 262

Darius Braziunas and Craig Boutilier. Stochastic local search for POMDP controllers. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, pages 690–696, San Jose, CA, 2004. 69, 170, 265

Darius Braziunas and Craig Boutilier. Local utility elicitation in GAI models. In *Proceedings of the Twenty-first Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 42–49, Edinburgh, 2005. 4, 7, 10, 17, 34, 41, 48, 49, 52, 74, 124, 126, 129, 138, 141, 151, 167, 168, 259

Darius Braziunas and Craig Boutilier. Preference elicitation and generalized additive utility (nectar paper). In *Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI-06)*, Boston, 2006. 17

Darius Braziunas and Craig Boutilier. Minimax regret-based elicitation of generalized additive utilities. In *Proceedings of the Twenty-third Conference on Uncertainty in Artificial Intelligence (UAI-07)*, pages 25–32, Vancouver, 2007. 4, 17, 34, 41, 46, 172, 205, 224, 246, 252, 259

Darius Braziunas and Craig Boutilier. Elicitation of factored utilities. *AI Magazine*, 29(4): 79–92, 2008. 259

Darius Braziunas and Craig Boutilier. Assessing regret-based preference elicitation with the UTPREF recommendation system. In *Proceedings of the Eleventh ACM Conference on*

Electronic Commerce (EC'10), pages 219–228, Cambridge, MA, 2010. 14, 18, 41, 46, 50, 169, 231, 259, 264, 272

Jack S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 43–52, Madison, WI, 1998. 251, 264

David C. Brown and William P. Birmingham. Understanding the nature of design. *IEEE Expert*, 12(2):14–16, 1997. 57

Robin Burke. Interactive critiquing for catalog navigation in e-commerce. *Artificial Intelligence Review*, 18(3-4):245–267, 2002. 251, 263

Colin F. Camerer, George Loewenstein, and Matthew Rabin, editors. *Advances in Behavioral Economics*. Princeton University Press, Princeton, New Jersey, 2003. 13, 43, 263

Urszula Chajewska and Daphne Koller. Utilities as random variables: Density estimation and structure discovery. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-00)*, pages 63–71, Stanford, 2000. 41, 48, 52, 68, 123, 125, 167, 253, 262

Urszula Chajewska, Lise Getoor, Joseph Norman, and Yuval Shahar. Utility elicitation as a classification problem. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 79–88, Madison, WI, 1998. 37, 49, 169, 264

Urszula Chajewska, Daphne Koller, and Ronald Parr. Making rational decisions using adaptive utility elicitation. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-00)*, pages 363–369, Austin, TX, 2000. 4, 9, 37, 48, 49, 51, 68, 123, 141, 151, 167, 168

Urszula Chajewska, Daphne Koller, and Dirk Ormoneit. Learning an agent's utility function by observing behavior. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML-01)*, pages 35–42, Williamstown, MA, USA, 2001. 41

Vijay Chandru and John N. Hooker. *Optimization Methods for Logical Inference*. Wiley, New York, 1999. 186

Richard M. Cyert and Morris H. de Groot. Adaptive utility. In Maurice Allais and Ole Hagen, editors, *Expected Utility Hypothesis and the Allais Paradox*, pages 223–241. D. Reidel, Dordrecht, 1979. 9, 48, 167

Joseph G. D'Ambrosio and William P. Birmingham. Preference-directed design. *Journal for Artificial Intelligence in Engineering Design, Analysis and Manufacturing*, 9:219–230, 1995. 58, 59

Morris H. de Groot. Decision making with an uncertain utility function. In Bernt P. Stigum and Fred Wenstøp, editors, *Foundations of Utility and Risk Theory with Applications*, pages 371–384. D. Reidel, Dordrecht, 1983. 9, 48, 167

Rina Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pages 211–219, Portland, OR, 1996. 137

Rina Dechter. *Constraint Processing*. Morgan Kaufmann, San Francisco, 2003. 29, 184

Jon Doyle and Richmond H. Thomason. Background to qualitative decision theory. *AI Magazine*, 20(2):55–68, 1999. 28

Jon Doyle and Michael P. Wellman. Representing preferences as *ceteris paribus* comparatives. In *Proceedings of the AAAI Spring Symposium on Decision-Theoretic Planning*, pages 69–75, Stanford, 1994. 28

Jon Doyle, Yoav Shoham, and Michael P. Wellman. A logic of relative desire. In Z. W. Ras and M. Zemankova, editors, *Methodologies for Intelligent Systems 6*, pages 16–31. Springer-Verlag, Berlin, 1991. 28

Jean-Philippe Dubus, Christophe Gonzales, and Patrice Perny. Choquet optimization using GAI networks for multiagent/multicriteria decision-making. In *Proceedings of the First Conference on Algorithmic Decision Theory (ADT-09)*, pages 377–389, Venice, 2009a. 124

Jean-Philippe Dubus, Christophe Gonzales, and Patrice Perny. Fast recommendations using GAI models. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 1896–1901, Pasadena, California, 2009b. 124

Jean-Philippe Dubus, Christophe Gonzales, and Patrice Perny. Multiobjective optimization using GAI models. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 1902–1907, Pasadena, California, 2009c. 124

James S. Dyer. Interactive goal programming. *Management Science*, 19:62–70, 1972. 37

Yagil Engel and Michael Wellman. Multiattribute auctions based on generalized additive independence. *Journal of Artificial Intelligence Research*, 37:479–525, 2010. 124, 125, 253, 262

Yagil Engel and Michael P. Wellman. Generalized value decomposition and structured multiattribute auctions. In *Proceedings of the Eighth ACM Conference on Electronic Commerce (EC'07)*, San Diego, California, 2007. 4, 119, 124

Yagil Engel and Michael P. Wellman. Cui networks: A graphical representation for conditional utility independence. *Journal of Artificial Intelligence Research*, 31:83–112, 2008a. 37

Yagil Engel and Michael P. Wellman. Evaluating the benefits of generalized additive representation in a multiattribute auction setting. In *Fourth Workshop on Advances in Preference Handling, AAAI-08*, 2008b. 125, 253, 262

Peter H. Farquhar. Utility assessment methods. *Management Science*, 30(11):1283–1300, 1984. 38, 40

Peter C. Fishburn. *Decision and Value Theory*. Wiley, New York, 1964. 54, 136

Peter C. Fishburn. Independence in utility theory with whole product sets. *Operations Research*, 13(1):28–45, 1965. 3, 5, 32, 33, 86, 91

Peter C. Fishburn. Methods of estimating additive utilities. *Management Science*, 13(7):435–453, 1967a. 43

Peter C. Fishburn. Interdependence and additivity in multivariate, unidimensional expected utility theory. *International Economic Review*, 8:335–342, 1967b. 4, 16, 34, 36, 74, 75, 76, 77, 80, 91, 92, 113, 123, 126, 257, 260, 268

Peter C. Fishburn. *Utility Theory for Decision Making*. Wiley, New York, 1970. 21, 34, 74, 77, 123, 125, 253, 262

Peter C. Fishburn. Multiattribute utilities in expected utility theory. In David E. Bell, Ralph L. Keeney, and Howard Raiffa, editors, *Conflicting Objective in Decisions*, pages 172–196. Wiley, 1977. 43

Simon French. *Decision Theory*. Halsted Press, New York, 1986. 8, 11, 21, 38, 45, 46, 48, 65, 67

Simon French and David Ríos Insua. *Statistical decision theory*. Oxford University Press, New York, 2000. 8, 22

Mirco Gelain, Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. Elicitation strategies for soft constraint problems with missing preferences: Properties, algorithms and experimental studies. *Artificial Intelligence*, 174(3–4):270–294, 2010. 29

Soumyadip Ghosh and Jayant Kalagnanam. Polyhedral sampling for multiattribute preference elicitation. In *Proceedings of the Fourth ACM Conference on Electronic Commerce (EC'03)*, pages 256–257, San Diego, CA, 2003. 11, 12, 47, 50, 60

Christophe Gonzales and Patrice Perny. GAI networks for utility elicitation. In *Proceedings*

of the Ninth International Conference on Principles of Knowledge Representation and Reasoning (KR2004), pages 224–234, Whistler, BC, 2004. 4, 6, 34, 40, 97, 113, 124, 259

Christophe Gonzales, Patrice Perny, and Sergio Queiroz. Preference aggregation in combinatorial domains using gai-nets. In *Joint DIMACS-LAMSADE workshop on computer science and decision theory*, volume 6, pages 165–179, Paris, 2006. 124

Christophe Gonzales, Patrice Perny, and Sergio Queiroz. Preference aggregation with graphical utility models. In *Proceedings of the Twenty-third AAAI Conference on Artificial Intelligence (AAAI-08)*, pages 1037–1042, Chicago, 2008. 124

Paul E. Green and Vithala R. Rao. Conjoint measurement for quantifying judgmental data. *Journal of Marketing Research*, 8(3):355–363, 1971. 40, 62

Paul E. Green and V. Srinivasan. Conjoint analysis in consumer research: Issues and outlook. *Journal of Consumer Research*, 5(2):103–123, 1978. 5, 39, 62

Paul E. Green and V. Srinivasan. Conjoint analysis in marketing: New developments with implications for research and practice. *Journal of Marketing*, 54(4):3–19, 1990. 62

Shengbo Guo and Scott Sanner. Real-time multiattribute Bayesian preference elicitation with pairwise comparison queries. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS-10)*, Sardinia, Italy, 2010. 167, 168

John C. Harsanyi. Games with incomplete information played by Bayesian players. part I: The basic model. *Management Science*, 14:159–182, 1967. 9, 49, 167

John C. Harsanyi. Games with incomplete information played by Bayesian players. part II: Bayesian equilibrium points. *Management Science*, 14:320–334, 1968. 9, 49, 167

Gordon B. Hazen. Partial information, dominance, and potential optimality in multiattribute utility theory. *Operations Research*, 34(2):296–310, 1986. 54, 56, 136

Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):53, 2004. 251

Hillary A. Holloway and Chelsea C. White, III. Question selection for multiattribute decision-aiding. *European Journal of Operational Research*, 148:525–543, 2003. 12, 50, 60, 62

Ronald A. Howard and James E. Matheson, editors. *Readings on the Principles and Applications of Decision Analysis*. Strategic Decision Group, Menlo Park, CA, 1984. 38

Vijay S. Iyengar, Jon Lee, and Murray Campbell. Q-Eval: Evaluating multiple attribute items using queries. In *Proceedings of the Third ACM Conference on Electronic Commerce*, pages 144–153, Tampa, FL, 2001. 11, 12, 47, 50, 59, 64

Ralph L. Keeney. *Value-Focused Thinking: A Path to Creative Decisionmaking*. Harvard University Press, Cambridge, MA, 1992. 125

Ralph L. Keeney and Howard Raiffa. *Decisions with Multiple Objectives: Preferences and Value Trade-offs*. Wiley, New York, 1976. 3, 5, 8, 21, 32, 33, 37, 38, 40, 41, 43, 114, 123, 125, 253, 255, 262

Craig W. Kirkwood and Rakesh K. Sarin. Ranking with partial information: A method and an application. *Operations Research*, 33(1):38–48, 1985. 54, 136

Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997. 251

Piero La Mura and Yoav Shoham. Expected utility networks. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 366–373, Stockholm, 1999. 37

Risto Lahdelma, Joonas Hokkanen, and Pekka Salminen. SMAA - stochastic multiobjective acceptability analysis. *European Journal of Operational Research*, 106:137–143, 1998. 47

S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, B 50(2):157–224, 1988. 137

Jordan J. Louviere, David A. Hensher, and Joffre D. Swait. *Stated Choice Methods: Analysis and Application*. Cambridge University Press, Cambridge, 2000. 7, 40

Andreu Mas-Colell, Micheal D. Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, New York, 1995. 41

George L. Nemhauser and Laurence A. Wolsey. *Integer Programming and Combinatorial Optimization*. Wiley, New York, 1988. 191

Andrew Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-00)*, pages 663–670, Stanford, CA, 2000. 41

Relu Patrascu, Craig Boutilier, Rajarshi Das, Jeffrey O. Kephart, Gerald Tesauro, and William E. Walsh. New approaches to optimization and utility elicitation in autonomic computing. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, pages 140–145, Pittsburgh, 2005. 46, 50, 70, 71, 72

Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, Reading, MA, 1984. 197, 199

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, 1988. 107

Bart Peintner, Paolo Viappiani, and Neil Yorke-Smith. Preferences in interactive systems: technical challenges and case studies. *AI Magazine*, 29(4):13–24, 2008. 13

David Portabella Clotet and Martin Rajman. Explicit trade-off and prospective analysis in electronic catalogs. In *ECAI Workshop on Recommender Systems*, Riva del Garda, Italy, 2006. 169, 239, 264

Steve Prestwich, Francesca Rossi, Brent Venable, and Toby Walsh. Constraint-based preferential optimization. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, Pittsburgh, PA, 2005. 29

Pearl Pu and Li Chen. User-involved preference elicitation for product search and recommender systems. *AI Magazine*, 29(4):93–103, 2008. 13

Pearl Pu and Boi Faltings. Decision tradeoff using example-critiquing and constraint programming. *Constraints*, 9(4):289–310, 2004. 263

Pearl Pu, Boi Faltings, and Marc Torrens. User-involved preference elicitation. In *IJCAI-03 Workshop on Configuration*, Acapulco, 2003. 13, 43, 169, 228, 254, 263

Kevin Regan and Craig Boutilier. Regret-based reward elicitation for Markov decision processes. In *Proceedings of the Twenty-fifth Conference on Uncertainty in Artificial Intelligence (UAI-09)*, pages 454–451, Montreal, 2009. 46

James Reilly, Jiyong Zhang, Lorraine McGinty, Pearl Pu, and Barry Smyth. Evaluating compound critiquing recommenders: a real-user study. In *Proceedings of the Eighth ACM Conference on Electronic Commerce (EC'07)*, pages 114–123, San Diego, California, 2007. 252, 263

Jason Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the Twenty-second International Conference on Machine Learning (ICML-05)*, 2005. 251

Francesca Rossi and Alessandro Sperduti. Acquiring both constraint and solution preferences in interactive constraint systems. *Constraints*, 9(4):311–332, 2004. 30

Francesca Rossi, Kristen Brent Venable, and Toby Walsh. *A Short Introduction to Preferences: Between AI and Social Choice*. Morgan and Claypool, 2011. 29

Thomas L. Saaty. A scaling method for priorities in hierarchical structures. *Journal of Mathematical Psychology*, 15:234–281, 1977. 65

Thomas L. Saaty. *The analytic hierarchy process: planning, setting priorities, resource allocation*. McGraw-Hill, 1980. 65

Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems 20 (NIPS-07)*, 2007. 264

Ahti Salo and Raimo P. Hämäläinen. Preference programming through approximate ratio comparisons. *European Journal of Operational Research*, 82:458–475, 1995. 67

Ahti Salo and Raimo P. Hämäläinen. Preference ratios in multiattribute evaluation (PRIME)–elicitation and decision procedures under incomplete information. *IEEE Trans. on Systems, Man and Cybernetics*, 31(6):533–545, 2001. 11, 37, 45, 47, 67, 227, 252

Ahti Salo and Raimo P. Hämäläinen. Preference programming. Manuscript available at http://www.sal.hut.fi/Publications/pdf-files/msal03b.pdf, 2004. 11, 45, 67

Ahti A. Salo and Raimo P. Hämäläinen. On the measurement of preferences in the Analytic Hierarchy Process. *Journal of Multi-Criteria Decision Analysis*, 6:309–319, 1997. 67

Paul A. Samuelson. A note on the pure theory of consumers' behavior. *Economica*, 5(17): 61–71, 1938. 7

Paul A. Samuelson. Consumption theory in terms of revealed preference. *Economica*, 15(60): 243–253, 1948. 7

Rakesh K. Sarin. Screening of multiattribute alternatives. *Omega*, 5(4):481–489, 1977. 54, 136

Leonard J. Savage. The theory of statistical decision. *Journal of the American Statistical Association*, 46(253):55–67, 1951. 11, 45, 176

Leonard J. Savage. *The Foundations of Statistics*. Wiley, New York, 1954. 21

Herbert A. Simon. A behavioral model of rational choice. *Quarterly Journal of Economics*, 69:99–118, 1955. 13, 43, 263

Edward A. Sykes and Chelsea C. White, III. Multiobjective intelligent computer-aided design. *IEEE Transactions on Systems, Man and Cybernetics*, 21(6):1498–1511, 1991. 57, 58

Sek-Wah Tan and Judea Pearl. Specification and evaluation of preferences for planning under uncertainty. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, pages 530–539, Bonn, 1994. 28

Olivier Toubia, Duncan I. Simester, John R. Hauser, and Ely Dahan. Fast polyhedral adaptive conjoint estimation. *Marketing Science*, 22(3):273–303, 2003. 12, 64, 65

Olivier Toubia, John R. Hauser, and Duncan I. Simester. Polyhedral methods for adaptive choice-based conjoint analysis. *Journal of Marketing Research*, 41(1):116–131, 2004. 11, 12, 40, 47, 50, 64, 252

Amos Tversky and Daniel Kahneman. Judgment under uncertainty: Heuristics and biases. *Science*, 185(4157):1124–1131, 1974. 13, 43, 263

Paolo Viappiani and Craig Boutilier. Regret-based optimal recommendation sets in conversational recommender systems. In *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys09)*, pages 101–108, New York, 2009. 205, 228, 252, 253, 263

Paolo Viappiani and Craig Boutilier. Optimal bayesian recommendation sets and myopically optimal choice query sets. In *Advances in Neural Information Processing Systems 23 (NIPS)*, Vancouver, 2010. 167, 168, 205, 228, 252, 253, 263

Paolo Viappiani, Boi Faltings, and Pearl Pu. Preference-based search using example-critiquing with suggestions. *Journal of Artificial Intelligence Research*, 27:465–503, 2006. 239, 252, 263

John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, 2nd edition, 1947. 3, 21, 24, 255

A. Wald. *Statistical Decision Functions*. Wiley, New York, 1950. 11, 45

Tianhan Wang and Craig Boutilier. Incremental utility elicitation with the minimax regret decision criterion. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 309–316, Acapulco, 2003. 12, 41, 46, 49, 50, 70, 168, 176, 227, 228, 252, 265

Martin Weber. Decision making with incomplete information. *European Journal of Operational Research*, 28:44–57, 1987. 9, 48, 54, 56, 136, 167

Michael P. Wellman and Jon Doyle. Preferential semantics for goals. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 698–703, Anaheim, 1991. 28

Chelsea C. White, Shigeru Dozono, and William T. Scherer. An interactive procedure for aiding multiattribute alternative selection. *Omega*, 11(2):212–214, 1983. 40, 54, 136, 266

Chelsea C. White, III, Andrew P. Sage, and Shigeru Dozono. A model of multiattribute decision-making and trade-off weight determination under uncertainty. *IEEE Transactions on Systems, Man and Cybernetics*, 14(2):223–229, 1984. 37, 40, 54, 136, 252, 267

Bob Wielinga and Guus Schreiber. Configuration-design problem solving. *IEEE Expert*, 12 (2):49–56, 1997. 57